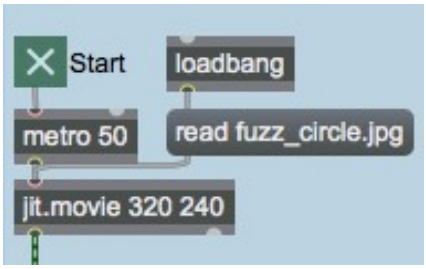
18-Processus itératifs et ré-échantillonnage de matrice

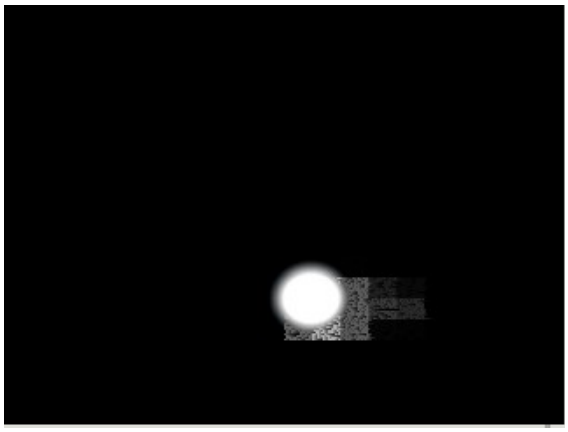
Ce didacticiel présente un exemple plus complexe d'utilisation des objets *jit.matrix* nommés, ainsi que de la manière d'utiliser des objets *jit.matrix* pour ré-échantillonner et sous-échantillonner une image.

Le coin supérieur gauche du patch contient un objet *jit.movie* dans lequel une image fixe (le fichier ***fuzz_circle.jpg***) est chargée à l'ouverture du patch.



Lire l'image

• Démarrez l'objet *metro* en cliquant sur le *toggle* située au-dessus de lui. Vous devriez voir une image apparaître dans la fenêtre *jit.p* en bas à droite du patch du tutoriel:



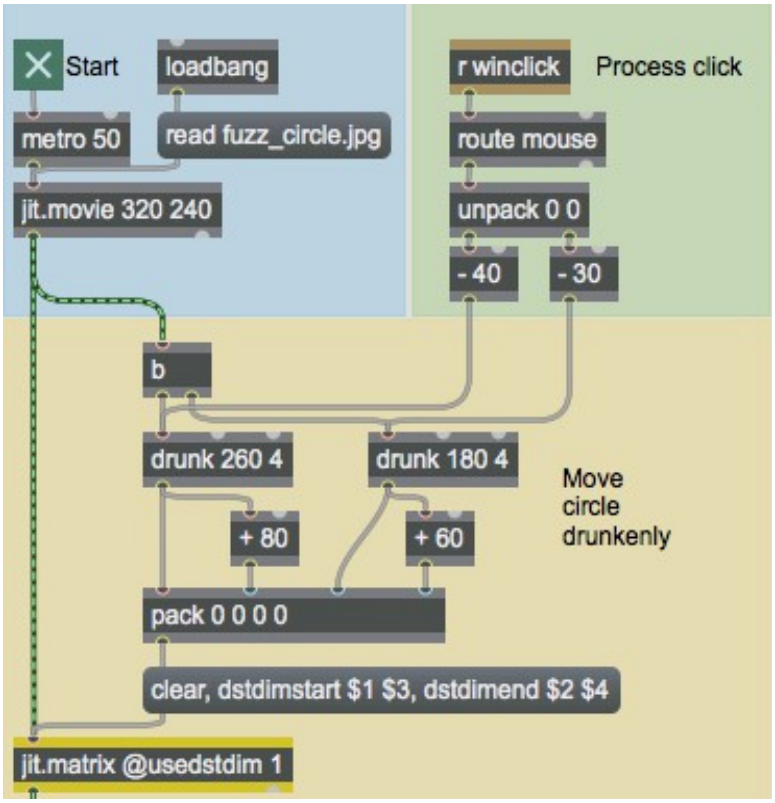
Notre petite comète

Le fichier ***fuzz_circle.jpg*** contient l'image d'un cercle blanc sur fond noir, dont la taille est mise à l'échelle pour apparaître comme un petit cercle à l'intérieur de notre matrice finale.

Le vrai ***fuzz_circle***.

Drunk

La partie supérieure du patch écrit l'image du fichier *jit.movie* dans le premier objet *jit.matrix* de la chaîne. Un **bang** généré par l'objet *bangbang* modifie les attributs **dstdimstart** et **dstdimend** de l'objet *jit.matrix* avec chaque image, faisant varier aléatoirement les coordonnées en utilisant les objets *drunk* de Max. Notez que notre premier objet *jit.matrix* à son attribut **usedstdim** défini à **1**, de sorte qu'il mettra à l'échelle la matrice d'entrée:



La partie drunk du patch

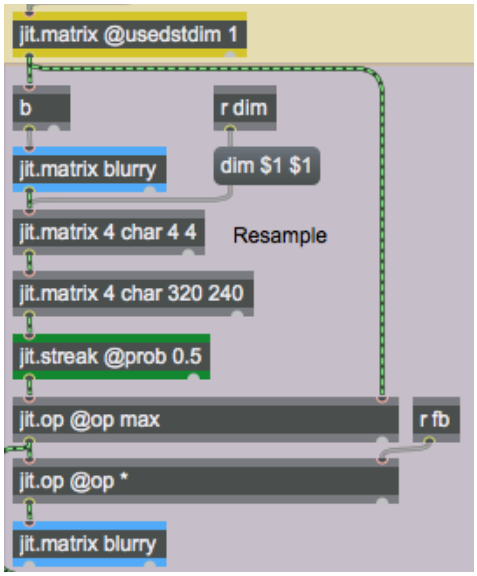
Ce premier *jit.matrix* sert donc simplement à mettre à l'échelle l'image du cercle pour qu'elle tienne dans une petite région (80 sur 60) de notre matrice de sortie. Notez que la boîte de *message* qui formate les coordonnées de l'image mise à l'échelle efface également la matrice à chaque image (avec un message **clear**), afin qu'il n'y ait pas d'artefacts d'une image précédemment écrite. Les objets Max *drunk* font varier le placement du cercle, ce qui le fait osciller (sans jeu de mots).

• Cliquez n'importe où dans la fenêtre *jit.p* dans le coin inférieur droit du patch. Le cercle sautera à la position où vous avez cliqué et commencera à se déplacer à partir de là.

Ce qui résulte d'un clic de souris dans la fenêtre *jit.pwindow* est envoyé à l'objet *receive* sous le nom d e **winclick**. Ce message est ensuite dépouillé de son sélecteur (souris) et les deux premiers éléments (la position x et y du clic de souris) sont extraits par l'objet *unpack*. Ces coordonnées sont ensuite utilisées pour définir la nouvelle origine des objets *drunk*.

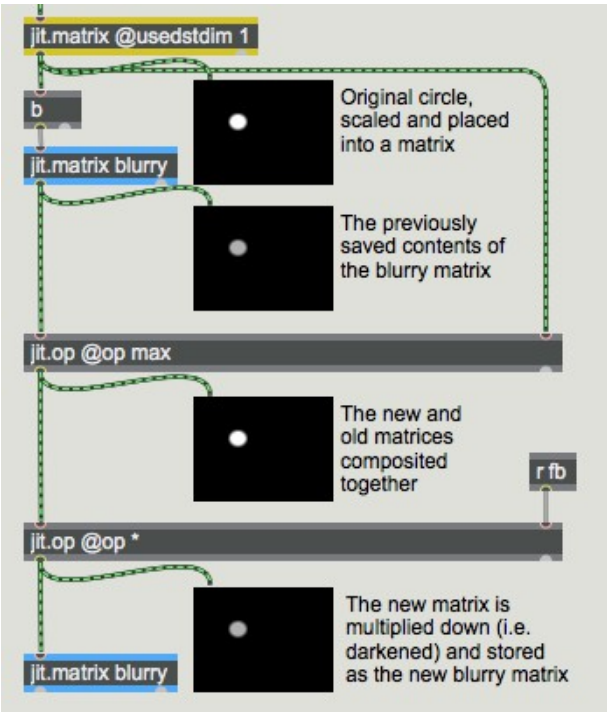
Le réseau de rétroaction

Une fois que notre image de cercle a été mise à l'échelle et placée de manière appropriée par l'objet *jit.matrix*, notre patch entre dans une chaîne de rétroaction centrée sur une paire d'objets *jit.matrix* partageant une matrice nommée **blurry**:



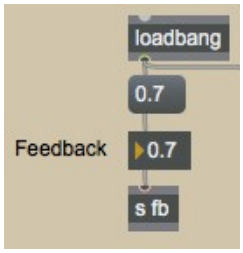
La boucle de feedback dans notre patch

Cette section du patch contient quatre objets *jit.matrix* (à l'exception de celui du haut qui réduit l'image du cercle). Deux de ces objets partagent un nom (**blurry**) et sont utilisés simplement pour stocker et récupérer les matrices précédentes générées par le reste du patch. L'objet *jit.matrix* plus haut envoie sa matrice à l'entrée la plus à droite du premier objet *jit.op* du patch. De plus, il envoie un **bang** au premier objet *namedjit.matrix* à l'aide d'un objet *bangbang*, lui faisant sortir sa matrice stockée (appelée «**blurry**»). Cette matrice finit par se retrouver dans l'entrée gauche du *jit.op*, où elle est ensuite affichée (par le *jit.pwindow*) et multipliée par un nombre scalaire (le deuxième objet *jit.op*). Elle finit par écraser la matrice floue précédente (en entrant dans l'objet *jit.matrix* nommée, en bas). Sans vous soucier de ce que font les objets Jitter intermédiaires, vous pouvez voir que la matrice floue **blurry** contiendra une version de la "trame" précédente de notre image de cercle:



Une carte réduite et illustrée de notre patch

Les nouvelles et les anciennes images sont combinées par le premier objet *jit.op* à l'aide de l'opérateur **max**. L'opérateur **max** compare chaque cellule des deux matrices et retient la cellule ayant la valeur la plus élevée. Le deuxième objet *jit.op* (avec l'opérateur *) sert à assombrir notre image en la multipliant par un scalaire (défini par la boîte de *nombre* située à droite du patch qui est envoyé à l'objet de réception nommé **fb**):

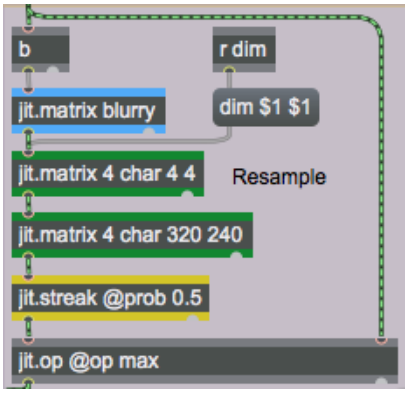


La quantité de feedback définit le degré d'obscurcissement de l'image avant son stockage dans la matrice ***blurry***.

• Modifiez la quantité de feedback du patch en jouant avec la boîte de *nombre* intitulée ***Feedback*** dans la région bleue du patch. Remarquez comment les tracés suivant le cercle augmentent ou diminuent lorsque vous déplacez le cercle en cliquant dans la fenêtre *jit.p*, en fonction du réglage de la quantité de rétroaction.

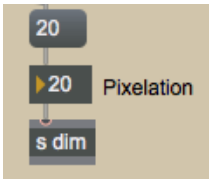
Sous-échantillonnage et sur-échantillonnage

La dernière étape de notre algorithme de traitement d'image concerne la partie du patch située entre le premier objet *jit.matrix* nommé, qui envoie la matrice sauvegardée lors de la précédente image par le *jit.matrix* du bas et le premier objet *jit.op*, qui compose la matrice précédente avec la nouvelle:



Utilisation d'objets *jit.matrix* pour le rééchantillonnage d'une image

Les deux objets *jit.matrix* de couleur verte dans le patch du didacticiel sont utilisés pour ***ré-échantillonner*** la matrice **blurry** qui sort de l'objet *jit.matrix* situé au-dessus d'eux. Le premier des deux objets *jit.matrix* a son attribut **dim** défini à **4 x 4** cellules. Cette taille peut être modifiée en définissant l'attribut avec la boîte de *nombres* dans la région bleue avec la légende ***Pixelation***. Ce nombre est envoyé à l'objet *receive* nommé **dim** au-dessus de l'objet *jit.matrix*.



Changer la pixellisation des pistes

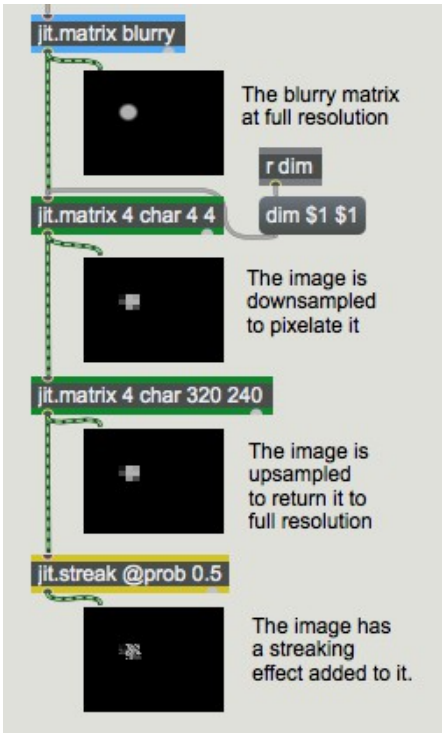
• Modifiez la boîte de *nombre* intitulée ***Pixelation*** dans la région bleue du patch du didacticiel. Remarquez comment les tracés des cercles changent.

En dé-échantillonnant la matrice de l'image, l'objet *jit.matrix* copie la matrice **320x 240** de son entrée dans une matrice beaucoup plus petite, en éliminant les données excédentaires. Le résultat est une pixellisation de l'image que vous pouvez contrôler avec le **dim** de la matrice.

Le second objet *jit.matrix* sur-échantillonne la matrice pour la ramener à une taille de **320x 240**. Ainsi, lorsque les objets Jitter suivants traiteront la matrice, ils disposeront d'une image à pleine résolution pour travailler et produiront une matrice à pleine résolution.

L'objet *jit.streak* ajoute un effet intéressant aux traces pixélisées en «striant» de manière aléatoire les cellules dans leurs voisines. L'attribut **prob** de *jit.streak* contrôle la probabilité qu'une cellule donnée de la matrice soit copiée sur une cellule voisine. Notre objet *jit.streak* a un attribut **prob** de **0.5**, ce qui signifie qu'il y a 50% de chances que cela se produise pour une cellule donnée.

> Détail technique: Par défaut, *jit.streak* copie les cellules vers la gauche. Changer l'attribut **direction** modifiera ce comportement. Il existe également un attribut **scale** qui détermine la luminosité des cellules "striées" par rapport à leurs valeurs d'origine. Le patch d'aide pour *jit.streak* explique plus en détail le fonctionnement de l'objet.



Notre chaîne d'effets avec des objets intermédiaires *jit.pwindow* pour montrer le traitement

Sommaire

Les paires d'objets *jit.matrix* nommés peuvent être utilisées efficacement pour stocker les itérations précédentes d'un processus Jitter. Ces techniques peuvent être utilisées pour générer des effets de retard vidéo en combinant la matrice précédente avec la matrice actuelle à l'aide d'objets de composition de matrice tels que *jit.op*. Vous pouvez également utiliser les objets *jit.matrix* pour ré-échantillonner une image (à l'aide de l'attribut **dim**), à la fois pour exécuter un algorithme plus efficacement (plus la matrice est petite, plus elle sera traitée rapidement par les images suivantes), ainsi que pour créer des effets de pixellisation. L'objet *jit.streak* exécute une répartition aléatoire dse cellules sur une matrice d'entrée en copiant des cellules sur leurs voisines selon un facteur de probabilité (défini par l'attribut **prob**).