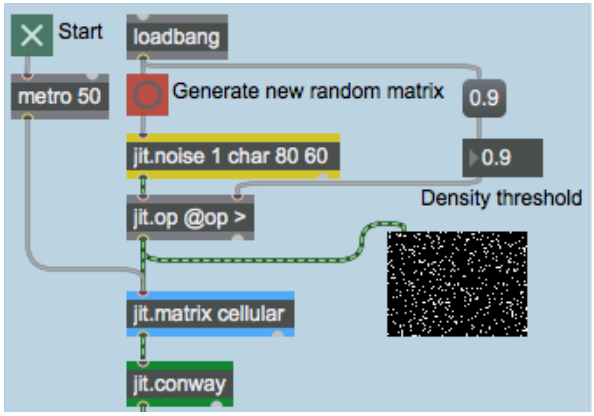
17-Feedback utilisant des matrices nommées

Ce tutoriel présente un exemple simple d'utilisation des objets *jit.matrix* nommés dans une boucle de rétroaction. Nous utiliserons une matrice de valeurs aléatoires pour amorcer un processus itératif (dans ce cas, le jeu de la vie de Conway).

Le patch du tutoriel génère une matrice initiale de valeurs aléatoires avec l'objet *jit.noise*:



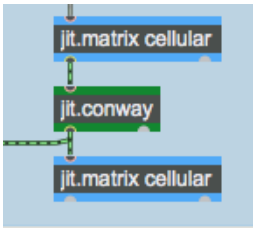
L'objet *jit.noise*.

L'objet *jit.noise* génère une matrice Jitter pleine de valeurs aléatoires. Les attributs **dim**, **planecount** et **type** de l'objet déterminent sa matrice de sortie (dans ce cas, nous voulons une matrice de 80 x 60 cellules de données **char** à un seul plan). Les valeurs aléatoires de nos cellules (qui sont initialement comprises entre 0 et 255) sont ensuite définies comme fausses (**0**) ou vraies (**255**) par l'objet *jit.op*. L'opérateur **>** de *jit.op* prend la valeur de la boîte de *nombres* (qui arrive à l'entrée droite de l'objet) et l'utilise comme opérateur de comparaison. Si la valeur de cellule est inférieure à cette valeur, la valeur de la cellule est fixée à **0**., sinon, la cellule est fixée à **255**. L'envoi d'un **bang** à *jit.noise* générera une nouvelle matrice aléatoire.

• Essayez de changer la boîte de *nombres* attachée à l'objet *jit.op*. Cliquez sur le *button* attaché à l'objet *jit.noise* pour générer une nouvelle matrice à chaque fois. Remarquez que des valeurs de comparaison plus élevées donnent moins de cellules blanches (**255**). La petite fenêtre *jit.p* sous l'objet *jit.op* vous montre la matrice aléatoire. Les données de la matrice à un plan sont correctement interprétées par l'objet *jit.pwindow* en tant que vidéo en niveaux de gris.

Rétroaction de la matrice Jitter

Le bruit quantifié que nous avons généré au sommet de notre patch va de l'objet *jit.op* à un objet *jit.matrix* portant le nom de **cellular**:



Deux objets nommés *jit.matrix* dans une boucle de rétroaction.

Cet objet *jit.matrix*, qui reçoit les messages **bang** d'un objet *metro* en haut du patch, est connecté à un objet appelé *jit.conway*, dont la sortie est reliée à une autre *jit.matrix* portant le même nom (**cellular**) que la première. Le résultat est que la sortie de l'objet *jit.conway* (quoi qu'il fasse) est écrite dans la même matrice que celle d'où provient son entrée, créant une boucle de rétroaction.

• Démarrez l'objet *metro* en cliquant sur le *toggle*. La fenêtre *jit.p* au bas du patch vous montrera la sortie de l'objet *jit.conway*.

Si vous souhaitez commencer avec une nouvelle matrice aléatoire, vous pouvez toujours copier une nouvelle matrice dans la boucle de rétroaction en cliquant sur le *button* associé à l'objet *jit.noise*. La matrice de l'objet *jit.op* ira dans notre matrice **cellular** partagée et sera utilisée dans la boucle de rétroaction.

«Game of Life»

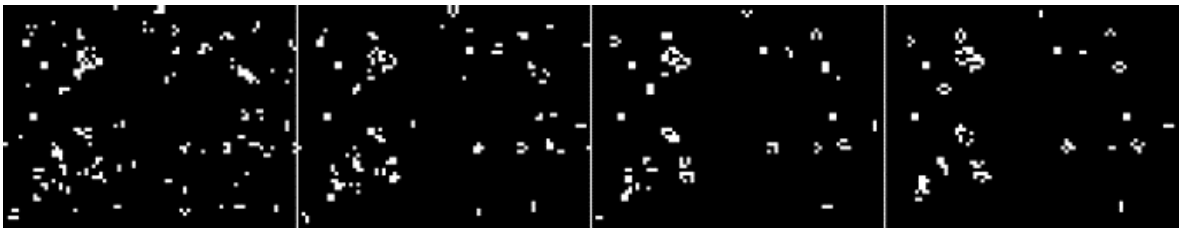
L'objet *jit.conway* exécute un algorithme d'automate cellulaire très simple appelé «Game of Life» sur une matrice d'entrée. Développé par John Conway à l'Université de Princeton, cet algorithme simule des cycles de survie organique dans un environnement où les réserves de nourriture sont limitées. Les cellules de la matrice sont considérées soit vivantes (**non-0**), soit mortes (**0**). Chaque cellule est comparée aux cellules qui l'entourent dans l'espace. Si une cellule vivante a deux ou trois voisines vivantes, elle reste en vie. Si elle a plus ou moins que ce nombre, elle meurt (c'est-à-dire est mis à **0**). Si une cellule morte a exactement trois voisins vivants, elle devient vivante (c'est-à-dire qu'elle prend la valeur **255**). C'est aussi simple que cela.

Chaque fois que l'objet *jit.conway* reçoit une matrice d'entrée, il exécute une génération du «Game of Life» sur cette matrice. Par conséquent, il est logique d'utiliser l'objet à l'intérieur d'une boucle de rétroaction, afin de pouvoir visualiser plusieurs générations de l'algorithme effectuées sur le même ensemble initial de données.

Par exemple, la matrice aléatoire initiale:

Quelques valeurs de matrice aléatoires

génère les matrices suivantes dans les quatre premières itérations via l'objet *jit.conway*:



Les quatre premières générations du «Game of Life» réalisées sur le jeu de données ci-dessus

Après avoir ensemencé la boucle de rétroaction avec une matrice aléatoire, vous pouvez activer l'objet *metro* et regarder l'algorithme fonctionner! «Game of Life» est conçu de telle sorte que la matrice finira par se stabiliser, soit en un groupe d'unités cellulaires auto-oscillantes, soit en une matrice vide (un monde mort). Dans les deux cas, vous pouvez simplement introduire une nouvelle série de chiffres par un **bang** et tout recommencer.

Sommaire

Vous pouvez utiliser l'attribut **name** de l'objet *jit.matrix* pour créer des boucles de rétroaction dans votre traitement Jitter. En utilisant deux objets *jit.matrix* portant le même nom à l'une des extrémités de la chaîne d'objets, vous créez un patch dans lequel la sortie de la chaîne est écrite dans la même matrice Jitter que celle d'où provient son entrée. L'objet *jit.noise* génère des matrices de nombres aléatoires de n'importe quel type, **dim** ou **planecount**. L'objet *jit.conway*, qui fonctionne le mieux dans une telle boucle de rétroaction, effectue une automatisation cellulaire simple sur une matrice d'entrée.