2-Créer une matrice

Qu'est-ce qu'une matrice?

Ce chapitre de didacticiel présente quelques méthodes pour gérer des données numériques dans une ***matrice***.

Si vous avez des doutes sur lce qu'est une matrice, veuillez tout d'abord lire le chapitre intitulé *Qu'est-ce qu'une matrice?* dans la section Topics. Pour résumer, une matrice est un schéma permettant de stocker et de modifier de grands ensembles de données numériques en plaçant les valeurs dans une grille virtuelle. En stockant des données dans une matrice, nous pouvons facilement identifier une valeur particulière par son emplacement dans la grille, et nous pouvons modifier plusieurs valeurs à la fois en faisant référence à tout ou partie d'une matrice.

Dans le chapitre précédent du didacticiel, nous avons utilisé l'objet *jit.window* pour ouvrir une fenêtre et afficher le contenu d'une matrice sous forme de pixels colorés. La matrice affichée provenait de l'objet *jit.movie*, un objet qui remplit continuellement sa matrice avec l'image actuelle d'une vidéo QuickTime. Cependant, si *jit.window* affichait une vidéo, c'est uniquement parce que c'était le contenu de la matrice qu'on lui demandait d'afficher; mais en fait, toutes les valeurs numériques d'une matrice peuvent être visualisées de la même manière. Le patch d'exemple de ce didacticiel montrera un exemple encore plus simple qui devrait vous aider à renforcer votre compréhension de ce qu'est une ***matrice***, idée centrale de Jitter.

L'objet jit.matrix

Créez un espace de stockage 16x12 pour des valeurs uniques de 8 bits.

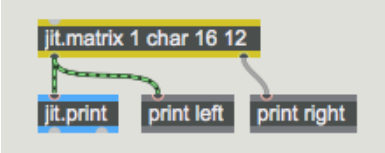
L'objet *jit.matrix* crée simplement une matrice, un espace de stockage en mémoire. Nous pouvons ensuite stocker et récupérer des valeurs numériques dans la matrice et nous pouvons utiliser d'autres objets pour imprimer les valeurs ou les afficher visuellement. Les arguments de *jit.matrix* sont un argument optionnel [**name**] (non inclus dans cet exemple), le [**planecount**] (combien de valeurs seront stockées dans chaque cellule de la matrice), le [**type**] de données (combien d'octets à utiliser pour représenter chaque nombre), et enfin le [**dim**] (abréviation de "dimensions", décrivant la taille de la matrice). Nous utilisons les crochets **[ ]** pour indiquer que ce n'est pas le mot réel que vous tapez comme argument, mais plutôt une description de la signification de l'argument. Dans cet exemple, l'objet va créer une matrice avec 1 ***plane*** (un nombre dans chaque emplacement de la matrice), en utilisant le type de données ***char*** (valeurs à un octet), avec des ***dimensions*** de 16 x 12 (ce qui équivaut à 192 cellules). Nous pouvons en déduire que la matrice est capable de contenir 192 valeurs numériques individuelles, chaque valeur allant de 0 à 255 (c'est la plage d'un seul octet).

> *Remarque*: nous décrivons toujours les dimensions d'une matrice bidimensionnelle au format x, y (largeur, hauteur), ce qui signifie que nous indiquons d'abord l'étendue de la dimension horizontale, puis la dimension verticale. Cela correspond à la manière dont ces dimensions sont généralement abordées dans la mise en page des écrans vidéo et informatiques (une image vidéo de 640x480, par exemple). Une autre façon de voir les choses est d'indiquer d'abord le nombre de ***colonnes*** (verticales) de données, puis le nombre

de ***lignes*** (horizontales). Toutefois, si vous appliquez les techniques matricielles de l'algèbre linéaire dans Jitter, vous voudrez peut-être noter que ce format (colonnes, lignes) est différent de la façon dont les matrices sont généralement décrites en algèbre linéaire, qui indique d'abord les lignes, puis les colonnes.

Nous avons connecté un objet *button* à l'entrée de *jit.matrix*. Vous vous souviendrez que les objets Jitter envoient leur nom de matrice lorsqu'ils reçoivent un message **bang** dans leur entrée gauche, donc ce *button* nous permettra de déclencher *jit.matrix* pour envoyer de son nom de matrice (sous la forme d'un message **jit_matrix**).

L'objet jit.print



Imprimer le contenu de *jit.matrix*.

Sous l'objet *jit.matrix* il y a un autre nouvel objet Jitter, *jit.print*. Cet objet accepte un nom de matrice (c'est-à-dire un message **jit_matrix**) dans son entrée et formate les valeurs de la matrice - dont le nombre peut être assez lourd - pour les imprimer dans la console Max. Il imprime les valeurs formatées directement dans la console Max, comme l'objet *print* de Max, puis transmet le nom de la matrice par sa sortie gauche dans un message **jit_matrix** (de la manière normale pour les objets Jitter).

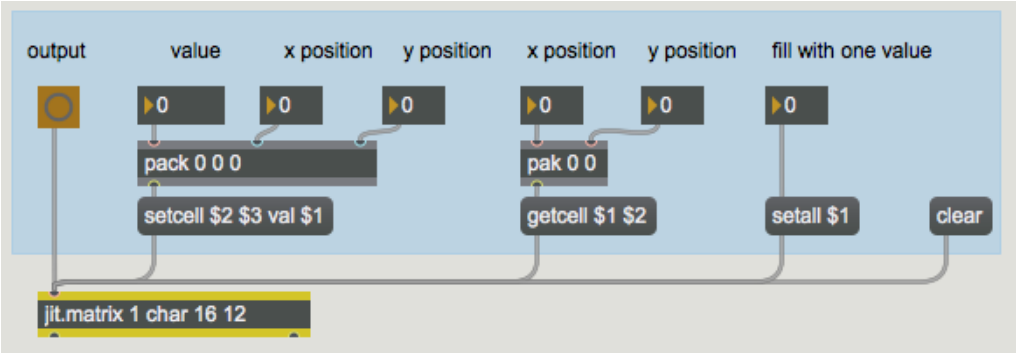
• Cliquez sur l'objet *button* marqué "output". Cela amène l'objet *jit.matrix* à communiquer son nom de matrice à *jit.print*, qui formate les valeurs et les imprime dans la console Max.

Dans la console Max, vous verrez maintenant le texte **left**: **jit_matrix [somename]**. Le mot **left** nous montre que ce texte a été imprimé par l'objet gauche *jit.print*, donc nous pouvons voir ce qui est ressorti de la sortie de *jit.matrix*. Parce que nous n'avons pas choisi de nom pour cette matrice (nous n'avons pas fourni de nom comme premier argument tapé sur *jit.matrix*), *jit.matrix* a assigné un nom de son propre choix. Il essaie de générer un nom unique qui n'est pas susceptible d'être utilisé ailleurs. Il choisit donc habituellement quelque chose d'étrange comme "u330000007". Dans ce cas, nous ne nous soucions pas vraiment du nom, mais il indique à l'objet *jit.print* quelle matrice de données formater pour l'impression.

En dessous, vous verrez toutes les valeurs de cette matrice particulière, soigneusement formatées en 16 colonnes et 12 lignes. Elles proviennent de *jit.print*. Elles sont toutes à **0** pour le moment, car nous n'avons encore rien placé dans la matrice.

Définition et interrogation de valeurs dans une matrice

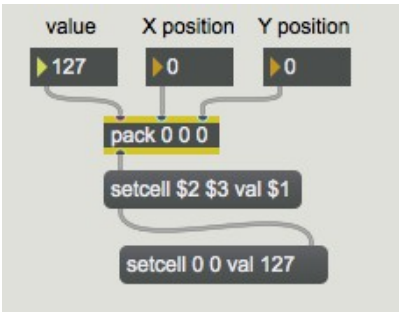
Dans le chapitre précédent, nous avons vu comment une matrice entière pouvait être remplie automatiquement avec les images de données couleur d'un film. Il est également possible de placer des valeurs numériques dans des cellules individuelles spécifiques de la matrice et de récupérer des valeurs à des endroits spécifiques. Les objets situés immédiatement au-dessus de *jit.matrix* dans cet exemple affichent quelques messages que vous pouvez utiliser pour définir et obtenir des valeurs dans une matrice.



Les messages **setcell** et **getcell** vous permettent d'accéder à des valeurs spécifiques dans une matrice.

Vous pouvez stocker des valeurs dans un emplacement particulier de la matrice à l'aide du message **setcell**. La syntaxe pour ce faire est: **setcell** [**cell coordinates**] **val** [**value(s)**]. Par exemple, le message **setcell 0 0 val 127** envoyé à notre *jit.matrix* définira la valeur de la toute première cellule de la matrice (c'est-à-dire la cellule située dans le coin supérieur gauche) à 127. Cela s'explique par le fait que nous numérotons les coordonnées des cellules dans chaque dimension en commençant par **0** et allant jusqu'à **1** de moins que l'étendue de cette dimension. Ainsi, dans cette matrice, les emplacements dans la dimension x sont numérotés de 0 à 15 et les emplacements dans la dimension y sont numérotés de 0 à 11. Ainsi, la cellule dans le coin inférieur droit serait décrite avec les coordonnées de cellule 15 11.

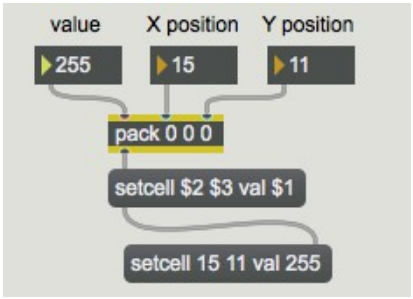
• La combinaison de l'objet *pack* **0 0 0** et de la boîte de *message* nous aide à formater un message **setcell** correct pour *jit.matrix*. Tout d'abord nous définissons les positions x et y où nous voulons stocker la valeur, puis nous spécifions une valeur à stocker à cet endroit. Avec les positions x et y à 0 0, utilisez la boîte de *nombre* étiquetée "valeur" pour envoyer le nombre 127 dans l'entrée gauche du pack 0 0 0. Le message **setcell 0 0 val 127** sera envoyé de la boîte de *message* à la *jit.matrix*.



Le message **setcell 0 0 val 127** définit la valeur de la position de cellule 0, 0 à 127.

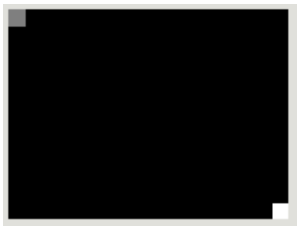
(S'il y avait plus d'un plan dans cette matrice, nous pourrions définir les valeurs dans un plan particulier d'une cellule ou dans tous les plans de la cellule. Cependant, cette matrice n'a qu'un seul plan, nous laisserons donc cela pour une autre fois.)

• Pour démontrer ce que nous avons dit précédemment sur la numérotation des positions de cellules, essayez d'envoyer le message **setcell 15 11 val 255** à *jit.matrix*. Saisissez d'abord le nombre **15** dans la boîte de *nombre* "x position" et le nombre **11** dans la "position en y", puis entrez le nombre **255** dans la boîte de *nombre* "valeur". Cliquez maintenant sur le bouton "output" pour voir comment la matrice a été modifiée. Une fois encore, la matrice entière sera imprimée dans la console Max via *jit.print*. Remarquez que les valeurs des cellules 0, 0 et 15, 11 ont été modifiées en 127 et 255.



Le message **setcell 15 11 val 255** définit la valeur de la position de cellule 15,11 à 255.

Dans la fenêtre Patcher, vous avez peut-être remarqué un changement dans la région rectangulaire noire. Les coins supérieurs gauche et inférieurs droit ont changé.



L'objet *jit.pwindow* affiche les valeurs numériques sous forme de couleurs (ou de niveaux de gris).

Cette région est un objet d'interface utilisateur appelé *jit.pwindow*. Dans la palette Ajouter un objet, cela ressemble à ceci:

L'icône *jit.pwindow* dans la palette d'objets

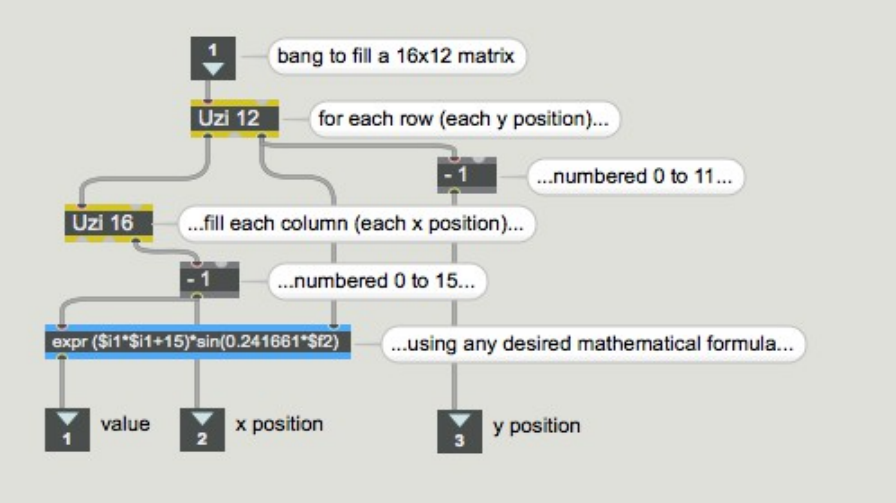
Lorsque vous cliquez sur cet objet dans la palette Ajouter un objet ou que vous le faites glisser dans la fenêtre du Patcher, il crée un petit objet rectangulaire. (Vous pouvez en augmenter la taille dans le coin inférieur droit de l'objet pour en ajuster les dimensions.) Cet objet est à peu près l'équivalent en fonction à l'objet *jit.window*, mais il affiche les données de la matrice dans la fenêtre du Patcher, plutôt que dans une fenêtre séparée.

Donc, ici, nous voyons littéralement l'affichage des valeurs numériques (dans ce cas, les données de la matrice dans la plage de 0 à 255) comme couleur. Comme il n'y a qu'un seul plan dans la matrice que nous affichons, l'affichage est monochrome, c'est-à-dire en niveaux de gris. Les valeurs 0 sont totalement noires et les autres valeurs représentent un certain niveau de gris pouvant aller jusqu'à 255 qui est totalement blanc. Ainsi, la valeur 255 de la cellule 15, 11 est affichée en blanc et la valeur 127 de la cellule 0, 0 est affichée en gris à 50%, à mi-chemin entre le noir et le blanc.

Vous pourriez dire: "Tout cela est très bien, mais il sera assez fastidieux de remplir une grande matrice de cette façon." Et vous auriez raison. Mais bien sûr, Max nous permet d'écrire une autre partie du programme qui automatisera le processus.

Remplir une matrice de manière algorithmique

• Double-cliquez sur l'objet patcher **fillmatrix** pour ouvrir la fenêtre de sub-patch ***fillmatrix***. Ce sub-patch génère 192 valeurs différentes, une pour chaque position dans la matrice, en introduisant différents nombres dans une expression mathématique.



Vous pouvez générer des valeurs de manière algorithmique pour remplir les cellules d'une matrice.

Lorsque l'objet *uzi* **12** reçoit un **bang** (du *button* "fil" dans la fenêtre principale du Patcher), il compte rapidement de 1 à 12 sur sa sortie droite et envoie un **bang** pour chaque compte dans sa sortie gauche. Ces **bang** déclenchent l'objet *uzi* **16**, de sorte qu'il envoie des nombres de 1 à 16 à chaque fois. Nous soustrayons 1 de ces nombres pour qu'ils aillent réellement de 0 à 11 et de 0 à 15, et nous utilisons les nombres résultants comme positions y et x dans la matrice. Pour chacune des 12 positions y, l'objet *uzi* **16** spécifie toutes les positions x, puis utilise ces nombres dans une expression mathématique (dans *expr*) pour calculer la valeur à stocker à cette position. Ces nombres sont envoyés aux sorties, et sont utilisés pour créer des messages **setcell** bien formés dans le patch principal, comme nous l'avions fait à la main précédemment.

L'expression mathématique ici est relativement peu importante. Il peut s'agir de n'importe quelle formule qui génère un modèle intéressant de données. Dans ce cas, nous avons choisi une formule qui produira une gradation sinusoïdale de la luminosité dans chaque colonne, et qui fera en sorte que la luminosité globale des colonnes augmente de gauche à droite (c.-à-d. lorsque **x** augmente).

• Fermez la fenêtre du sub-patch ***fillmatrix*** et cliquez sur le *button* "fil". La matrice est remplie de valeurs (générées par les objets *uzi* du sub-patch) en un seul clic du programmateur de Max. Cliquez maintenant sur le *button* "output" pour afficher le contenu de la matrice. Les valeurs numériques seront imprimées dans la console Max et affichées dans *jit.pwindow*.

Même pour une petite matrice 16x12 comme celle-ci, il est difficile pour nous de percevoir une tendance dans les données numériques juste en regardant une impression de chiffres dans la console Max. Cependant, l'affichage dans la fenêtre *jit.pwindow* nous donne une idée très claire et immédiate de la façon dont les valeurs varient dans la matrice. Cela démontre l'un des avantages de la visualisation des données numériques.

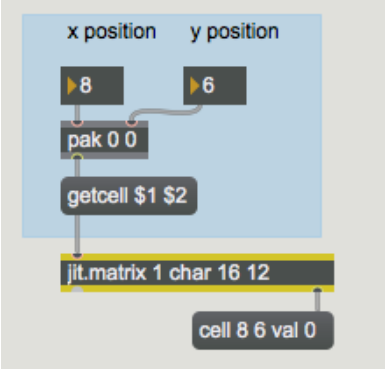
Vous pouvez sans aucun doute imaginer d'autres façons de remplir une matrice de façon algorithmique dans Max, et en fait nous démontrerons d'autres façons dans les chapitres suivants du tutoriel.

Autres messages à *jit.matrix*

Il y a beaucoup d'autres messages compris par *Jit.matrix*, plus que ce que nous pouvons démontrer complètement ici. Sur le côté droit du Patcher, nous montrons quelques autres messages pratiques pour remplir instantanément une *jit.matrix* avec toutes les mêmes valeurs. Le message **clear** envoyé à *jit.matrix* définit toutes ses valeurs à **0** et le message **setall** (le mot **setall** suivi d'une valeur) définit chaque position dans la matrice à cette valeur.

Nous démontrons également le message **getcell**. Le mot **getcell** suivi d'un emplacement dans la matrice (positions x et y) fera en sorte que *jit.matrix* envoie les coordonnées et la valeur de la cellule de cette position par sa sortie droite.

• Entrez une valeur y, puis une valeur x dans les boîtes de *nombress* au-dessus de la boîte de *message* **getcell $ 1 $ 2**, et observez ce qui est imprimé dans la console Max. Notez que la valeur de cette position de la matrice est envoyée par la sortie droite de *jit.matrix*.



Interrogez la ou les valeurs à la position de matrice 8, 6; rapporte **cell 8 6 val [value(s)]**

Dans les prochains chapitres du didacticiel, vous verrez différentes façons d'utiliser les valeurs extraites d'une matrice.

Sommaire

L'objet *jit.matrix* crée un espace de stockage pour une matrice de données nommée, avec les dimensions, les plans et le type de données que vous spécifiez. Cette matrice peut être remplie avec les données provenant d'un autre objet Jitter (tel que *jit.movie*), ou par des messages tels que **setall [value]** pour définir la valeur dans toutes les cellules ou **setcell [position] val [value (s)]** pour définir une cellule spécifique. Vous pouvez utiliser un algorithme ailleurs dans le patch pour remplir la matrice en fonction d'une formule ou d'un ensemble de règles.

Pour obtenir les données d'une cellule spécifique, vous pouvez utiliser le message **getcell [position]**. Pour voir toutes les données numériques imprimées dans la console Max, utilisez l'objet *jit.print* pour formater les données de la matrice et les imprimer. Pour voir les données de la matrice affichées en couleurs, utilisez l'objet *jit.pwindow*. Ceci est similaire à l'utilisation de l'objet *jit.window* présentée dans le didacticiel 1.

Dans ce tutoriel, nous avons visualisé des données numériques que nous avons générées nous-mêmes, plutôt que des vidéos numériques, comme dans le chapitre précédent. Le principe de stockage est le même dans les deux cas. Qu'il s'agisse d'une matrice utilisée pour stocker des informations de couleur pour chaque pixel d'une image de vidéo numérique, ou de nombres abstraits que nous souhaitons visualiser sous forme de couleur, les nombres des deux chapitres sont stockés dans une matrice bidimensionnelle et sont facilement affichés facilement dans une *jit. .window* ou *jit.pwindow*.