5-Couleur ARGB

La couleur dans Jitter

Dans ce chapitre, nous verrons comment la couleur est gérée dans Jitter. Nous nous concentrerons sur la représentation numérique d'une couleur, et inversement sur la visualisation des nombres en tant que couleur. Une discussion approfondie de la théorie des couleurs - savoir comment la lumière et la matière produisent notre sensation de couleur - va bien au-delà de la portée de ce didacticiel, tout comme une discussion sur les nombreuses façons de représenter les informations de couleur numériquement. Si vous souhaitez en savoir plus sur la théorie des couleurs et/ou les représentations numériques de la couleur, vous pouvez trouver d'autres sources d'informations répertoriées dans la ***bibliographie***.

Nous présentons ici une méthode de représentation de la couleur et la façon dont cette représentation est traitée dans les matrices de Jitter.

Composants de couleur: RVB

Il est possible de produire n'importe quelle couleur souhaitée en mélangeant trois couleurs de lumière - le rouge, le vert et le bleu - chacune avec sa propre intensité. C'est ce que l'on appelle la synthèse additive, qui consiste à générer une couleur en ajoutant des quantités uniques de trois couleurs "primaires" de la lumière. (L'inverse est la synthèse soustractive: mélanger des pigments colorés, comme la peinture, qui absorbent certaines couleurs de la lumière et réfléchissent les autres.) Ainsi, nous pouvons décrire toute lumière colorée en fonction de son intensité aux trois fréquences qui correspondent à la couleurs spécifiques rouge, vert et bleu.

Dans Jitter, c'est la manière la plus courante de décrire une couleur: comme une combinaison d'intensités exactes du rouge, du vert et du bleu. Pour chaque pixel d'une image - qu'il s'agisse d'une vidéo, d'une photo ou de toute autre matrice 2D - nous avons besoin d'au moins trois valeurs, une pour chacune des trois couleurs de base. Par conséquent, pour les images couleur à l'écran, nous utilisons le plus souvent une matrice 2D avec au moins trois plans.

Le canal alpha

Un quatrième plan est souvent utile pour ce que l'on appelle le ***canal alpha*** - un canal qui stocke des informations sur le degré de transparence d'un pixel lorsqu'il est superposé à une autre image. Nous ne traiterons pas spécifiquement du canal alpha dans ce chapitre du didacticiel, mais nous le mentionnons ici car son inclusion est standard dans la plupart des objets Jitter qui traitent spécifiquement des représentations de la couleur. Dans la plupart des cas, le canal alpha est stocké dans le premier plan (qui est le plan 0, car les plans d'une matrice sont numérotés à partir de 0) et les valeurs RVB sont dans les plans 1, 2 et 3.

Données de couleur: char, long ou float

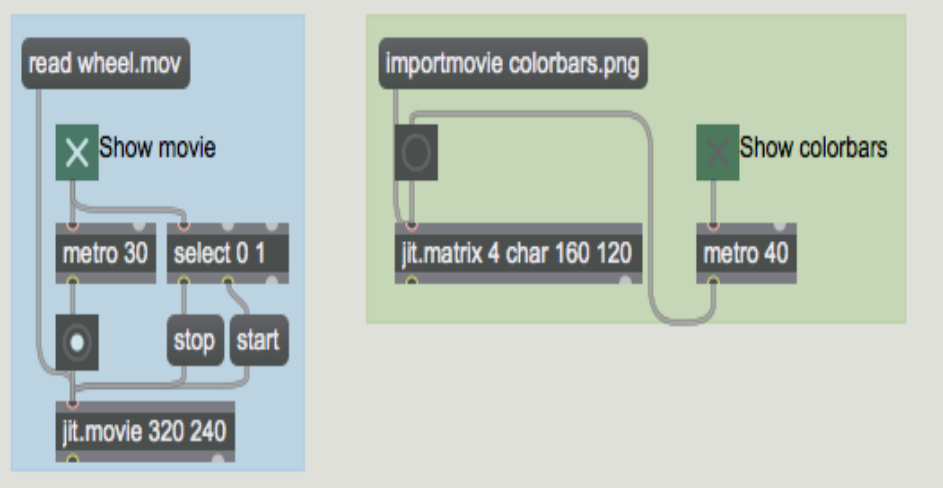
Dans les applications informatiques, il est assez courant d'utiliser 8 bits d'information par valeur de couleur de base. 8 bits nous donnent la possibilité d'exprimer 256 (de la puissance 2 à la 8) valeurs différentes. Cela signifie que si nous utilisons 8 bits pour le rouge, 8 pour le vert et 8 pour le bleu, nous pouvons exprimer 16 777 216 (224) couleurs différentes. C'est une variété de couleurs suffisante pour couvrir à peu près toutes les nuances que nous sommes capables de distinguer.

Donc, si nous n'avons besoin que de 8 bits de résolution pour représenter chaque valeur de couleur de base, cela signifie que le type de données ***char*** en 8 bits est suffisant pour représenter la valeur dans chaque plan d'une matrice à quatre plans d'informations sur les couleurs. Nous pourrions utiliser les types de données ***long***, ***float32*** ou ***float64***, et Jitter nous le permettra certainement, mais nous utiliserions des types de données beaucoup plus grands que ce dont nous avons réellement besoin. Etant donné qu'une image vidéo plein écran peut contenir un très grand nombre de pixels (une image 640x480 comporte 307 200 pixels), il est souvent plus judicieux – afin de conserver de la mémoire et d'accélérer le traitement - d'utiliser le type de données ***char***.

Lorsqu'une matrice contient des données ***char*** de 8 bits, on peut considérer que ces 8 bits représentent des nombres compris entre 0 et 255, ou qu'ils représentent des gradations entre 0 et 1 (c'est-à-dire un nombre fractionnaire à virgule fixe). Lorsqu'un objet Jitter contenant des données ***char*** reçoit des valeurs numériques d'autres objets Max, il s'attend généralement à les recevoir sous la forme de flottants dans la plage 0 à 1. Il effectuera les calculs internes nécessaires pour convertir un flottant de Max en valeur ***char*** appropriée. (Il y a quelques exceptions à cette règle. Par exemple, l'objet *jit.op* peut accepter soit des flottants dans la plage 0 à 1 soit des *ints* dans la plage 0-255 dans son entrée droite, comme indiqué dans le didacticiel 3.) Pour en savoir plus sur l'utilisation du type de données ***char*** dans les matrices de Jitter, voir le tutoriel *Qu'est-ce qu'une matrice*?

Isolation des plans d'une matrice

En haut du patch, on vous donne deux matériaux sources colorés différents. L'un est la vidéo d'un jeu d'arcade, et l'autre est l'ensemble standard de barres de couleur à utiliser pour l'étalonnage vidéo. Vous pouvez choisir de voir l'un ou l'autre en activant l'un des objets *metro* (faire des **bangs** de manière répétée sur l'objet contenant la matrice que vous voulez voir).



Visionner une vidéo ou une image fixe

• Cliquez sur le bouton marqué "Afficher le film" au-dessus de l'objet *metro***30** pour visualiser la vidéo.

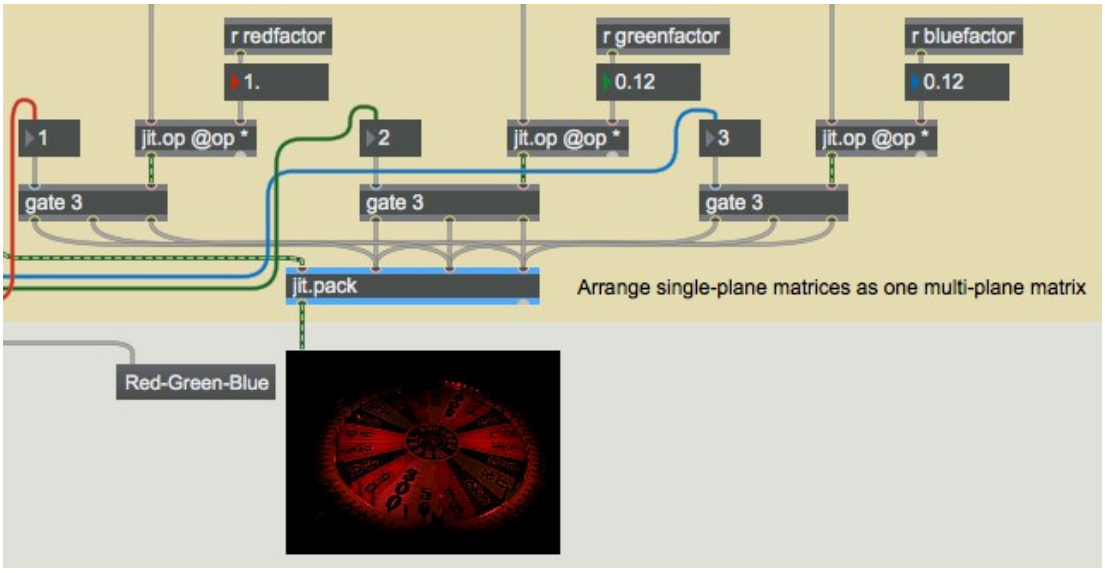
Cet exemple de patch décompose une matrice à 4 plans d'informations de couleur en quatre matrices à 1 plan, afin de vous permettre de voir - et de modifier - chaque plan individuellement. Nous y parvenons avec un objet appelé *jit.unpack*. Tout comme l'objet Max *unpack* décompose une liste en nombres individuels, *jit.unpack* décompose une matrice multiplan en matrices 1-plan individuelles. Vous pouvez saisir un argument pour indiquer à *jit.unpack* le nombre de plans à attendre dans la matrice entrante, mais par défaut, il s'attend à quatre plans, puisque c'est la norme pour les données de couleur. Nous sommes intéressés à voir le contenu des plans rouge, vert et bleu.

Donc nous envoyons donc les plans 1, 2 et 3 à des objets *jit.pwindow* individuels. Dans ce cas, le canal alpha ne nous intéresse pas, donc nous ne prenons donc pas la peine d'afficher le plan 0.

Dépaqueter une matrice multi-plan en tant que matrices mono-plan

Vous pouvez voir ici le contenu de chaque plan de couleur, sous la forme de trois images monochromes. Les pixels plus clairs indiquent des valeurs plus élevées pour cette couleur. En envoyant chaque matrice à un objet *jit.op*, nous obtenons un contrôle individuel sur la force de chaque couleur, et pouvons modifier la balance des couleurs. Nous envoyons ensuite les matrices individuelles (modifiées) à un objet *jit.pack* pour les recombiner en une matrice à 4 plans, pour les afficher dans la fenêtre *jit.pwindow*.

• Essayez, par exemple, de réduire l'intensité du vert et du bleu à 0,12 pour créer une image plus rouge.

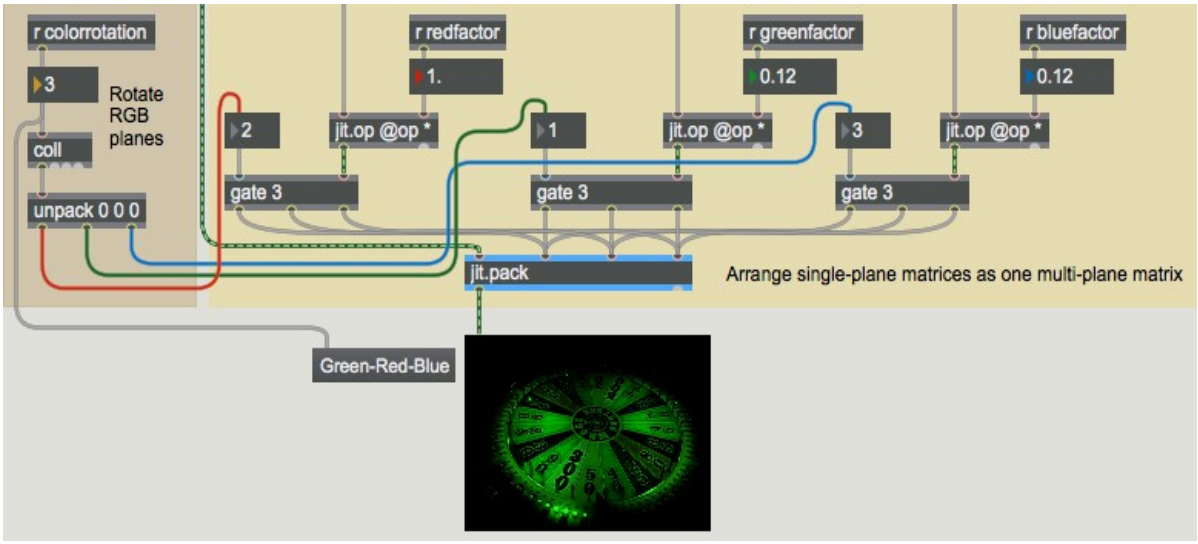


Réduire l'intensité de certaines couleurs pour modifier l'équilibre des couleurs

Rotation des couleurs

Pour illustrer une autre astuce de couleur, nous avons envoyé chaque plan de couleur à travers un objet *gate*, afin que chaque matrice puisse être acheminée vers une entrée différente (plan de couleur) de *jit.pack*. De cette façon, la signification de chaque plan peut être réaffectée et nous pouvons essayer toutes les permutations des affectations de couleurs possibles en choisissant un paramètre dans l'objet *coll* sur le côté gauche du patch.

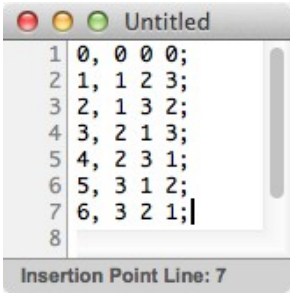
• Faites glisser la boîte de *nombre* marquée "Faire pivoter les plans RVB" pour essayer différentes réaffectations des trois plans de couleur. (Notez que le plan 0 est envoyé directement de *jit.unpack* à *jit.pack*; c'est le message **jit_matrix** arrivant dans l'entrée gauche de *jit.pack* qui déclenche la sortie de la matrice vers la fenêtre *jit.pwindow.*) Si vous choisissez l'élément 3 dans le *coll*, vous obtenez le résultat montré ci-dessous.



Les plans de couleur individuels sont réaffectés; les plans rouges et verts sont ici échangés.

L'exemple ci-dessus montre les plans vert et bleu originaux réduits par un facteur de 0,12 et les *gate*s sont configurées de manière à ce que les plans rouge et vert soient échangés lorsqu'ils sont envoyés à *jit.pack*, ce qui donne une image contenant davantage de vert. L'objet *coll* contient toutes les permutations possibles des affectations des plans RVB.

• Double-cliquez sur l'objet *coll* pour voir son contenu.



Permutations des assignations de plans RVB

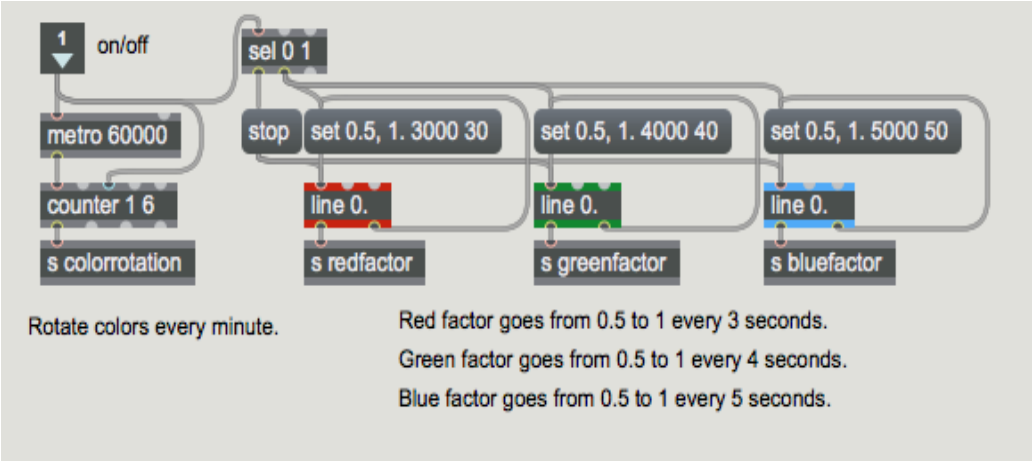
Les éléments de la **list** qui sortent de *coll* sont dépaquetés et envoyés pour allouer les sorties des trois objets *gate*. Le nombre envoyé à *coll* est également envoyé à un *umenu* (en mode Label) pour montrer la rotation des couleurs en mots - dans ce cas, "Vert-Rouge-Bleu".

Modifications de couleur automatisés

Pour un dernier exercice de modification des couleurs, nous avons mis au point un processus automatisé permettant de modifier en permanence l'échelle et la rotation des couleurs.

• Cliquez sur le *toggle* marqué "Automatiser les changements de couleur". Les facteurs de mise à l'échelle pour les trois plans de couleur changent tous continuellement. Double-cliquez sur l'objet de

patcher **patchercolorgames** pour voir le contenu du sub-patch.



sub-patch [colorgames]

Le sub-patch utilise des objets *line* pour envoyer des valeurs qui progressent de 0,5 à 1 pour chacun des facteurs d'échelle des couleurs. Le facteur rouge change au cours de 3 secondes, vert en 4 secondes et bleu toutes les 5 secondes. (Les trois objets *line* sont ainsi synchronisés donc une fois toutes les 60 secondes.) Toutes les 60 secondes, une nouvelle rotation de couleur est sélectionnée par la combinaison *metrocounter*.

• Fermez la fenêtre de sub-patch [***colorgames***]. Vous pouvez essayer toutes ces mêmes modifications de couleur sur un matériel source différent. Cliquez sur le bouton "Afficher le film" pour arrêter le *metro*. (Notez que nous utilisons également ce *toggle* pour démarrer et arrêter la lecture du film dans l'objet *jit.movie*. Il n'y a aucune raison de garder le film en lecture si nous ne le regardons même pas.) Cliquez maintenant sur le *toggle* marqué "Afficher les barres de couleur". pour afficher les barres de couleur. Expérimentez en changeant les facteurs d'échelle et de rotation sur cette image.

Sommaire

Lorsqu'une fenêtre *jit.window* ou *jit.pwindow* reçoit une matrice 2D à plan unique, elle affiche les valeurs sous la forme d'une image monochrome (échelle de gris). Lorsqu'elle reçoit une matrice 2D à 4 plans, elle interprète les plans comme des valeurs alpha, rouge, verte et bleue et affiche les couleurs résultantes en conséquence. Cette représentation ARGB dans une matrice à 4 plans est la manière la plus courante de représenter les couleurs dans Jitter.

Parce que chacune des couleurs de base ne nécessite que 8 bits de précision pour représenter sa gamme complète, il est courant dans Jitter d'utiliser le type de données **char** pour les données de couleur. Ainsi, la plupart des objets liés à QuickTime (tels que *jit.movie*) et nombreux objets spécifiquement conçus pour manipuler les couleurs (tels que *jit.brcosa* et *jit.colorspace*, illustrés dans les chapitres suivants du didacticiel) s'attendent à recevoir des matrices 2D à 4 plans de données **char**. (De nombreux autres objets s'adaptent facilement à d'autres types de données, cependant. Consultez la documentation de référence de l'objet en question en cas de doute.) Vous pouvez considérer que les données **char** représentent des valeurs fractionnaires comprises entre 0 et 255, ou qu'elles représentent des valeurs fractionnaires comprises entre 0 et 1. La plupart du temps, les objets qui contiennent des données **char** s'attendent à recevoir des valeurs numériques d'autres objets Max spécifiés comme des flottants dans la plage 0 à 1.

L'objet *jit.unpack* divise une matrice multiplan en matrices individuelles à plan unique. L'objet *jit.pack* combine les matrices à plan unique en une seule matrice à plans multiples. En traitant chaque plan séparément, vous pouvez contrôler la balance des couleurs d'une image, voire même réassigner la signification des plans individuels.