6-Ajuster les niveaux de couleur

jit.scalebias

Ce tutoriel développe la discussion sur les couleurs du chapitre précédent et introduit un objet spécialement conçu pour modifier les plans de couleurs ARVB d'une matrice: *jit.scalebias*.

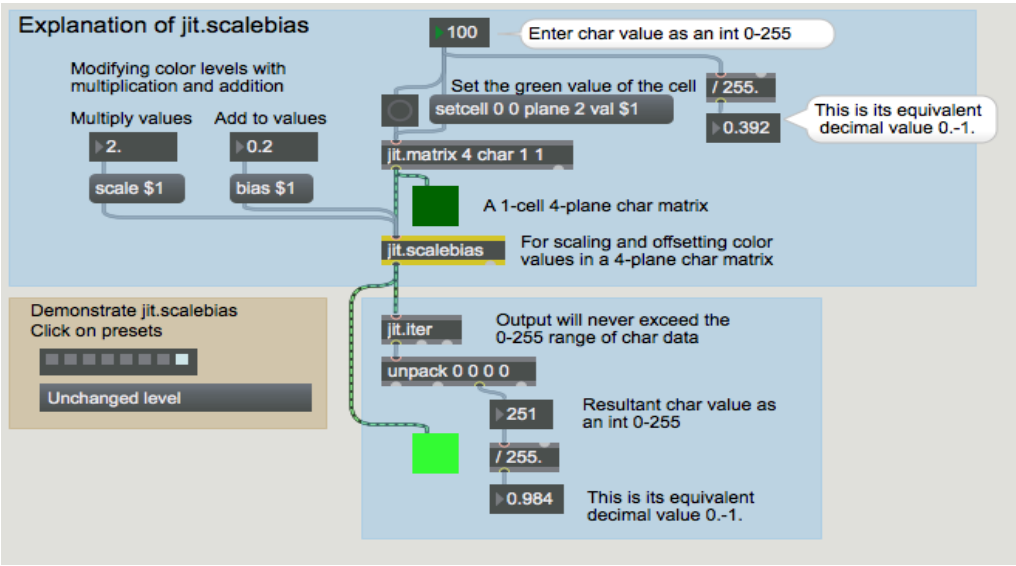
Le terme échelle dans ce cas fait référence au processus de mise à l'échelle de valeurs par un certain facteur; c'est-à-dire les multiplier par une quantité donnée. Le terme biais fait référence à la compensation des valeurs en leur ajoutant une quantité. En combinant les processus de multiplication et d'addition, vous pouvez obtenir une correspondance linéaire des valeurs d'entrée aux valeurs de sortie.

Comme *jit.scalebias* s'occupe de modifier les informations de couleur ARVB dans une image, il ne traite que les matrices à 4 plans du type de données **char**. (Voir le didacticiel 5 pour une discussion sur la couleur ARVB et les données **char**.)

Math avec données char

Comme indiqué dans le chapitre précédent, les données **char** 8 bits peuvent être représentées comme valeurs entières de 0 à 255 ou comme des valeurs fractionnelles de 0 à 1. Dans le didacticiel 2, par exemple, nous avons vu que l'objet *jit.print* rapporte des données **char** comme des valeurs entières de 0 à 255. Toutefois, dans de nombreux cas où nous modifions les valeurs **char** dans une matrice (en modifiant l'un de ses attributs), l'objet Jitter s'attend à recevoir la valeur de l'attribut sous forme de valeur flottante. Comme cela peut être un peu déroutant, nous avons fourni une démonstration dans ce patch de tutoriel.

• Ouvrez le patch du tutoriel. Cliquez sur l'objet de *patcher* **explain_scalebias** au milieu du patch pour afficher le contenu du sub-patch [***explain_scalebias***].



*Une démonstration de math **float** sur des données char.*

Dans l'exemple ci-dessus, nous avons créé une très petite matrice. Elle comporte 4 plans de données ***char***, mais elle ne comporte qu'une seule cellule. Cela nous permet de nous concentrer sur ce qu'il advient à seule valeur numérique dans la matrice. Vous pouvez voir que nous avons défini la valeur du plan 2 (plan vert) sur **100**. À gauche de l'exemple, vous pouvez voir comment cette valeur

numérique entière serait représentée par une valeur fractionnaire entre 0 et 1: dans ce cas, c'est **0,392**, ce qui représente 100/255 du chemin entre 0 et 1.

L'objet *jit.scalebias* multiplie les valeurs par une certaine quantité (spécifiée par l'attribut **scale**), et ajoutera ensuite une certaine quantité aux valeurs - spécifiée par l'attribut **bias**. Lorsque cette matrice est modifiée par l'objet *jit.scalebias*, toutes les valeurs seront traitées comme des flottants pour les besoins du calcul, puis seront ré-stockées dans la matrice comme des données ***char***.

Les attributs **scale** et **bias** sont définis ici avec les messages **scale 2.0** et **scale 0.2**. Le facteur d'échelle (le multiplicateur) est de 2,0 dans ce cas, et le biais (le décalage ajouté par la suite) est de 0,2. Donc, pour comprendre ce qui se passe dans *jit.scalebias*, nous devons considérer que la valeur verte est 0,392 fois 2,0 plus 0,2, ce qui donne à 0,984. L'objet *jit.iter* rapporte les valeurs dans chaque plan de chaque cellule (il n'y a qu'une seule cellule dans cette matrice), et nous pouvons voir que la valeur (lorsqu'elle est stockée dans la matrice sous forme de ***char***) est **251** (ou 0,984 sur une échelle de 0 à 1).

(Comme un exercice de calcul mental, pouvez-vous calculer les valeurs que *jit.scalebias* produira dans les plans rouge et bleu dans l'exemple ci-dessus? Puisque les valeurs de ces plans dans la matrice d'origine sont 0, les valeurs dans la matrice résultante seront 0 fois 2,0 plus 0,2, ce qui est égal à 0,2, qui est égal à 51 sur une échelle de 0 à 255. Donc les valeurs RVB affichées par l'objet *jit.pwindow* en bas sont donc 51 251 51.)

Plus d'exemples

Si l'explication précédente était claire comme de l'eau de roche pour vous, vous pouvez ignorer ces exemples supplémentaires. Mais au cas où vous ne comprendriez toujours pas encore très bien comment fonctionnent les opérations mathématiques avec les données ***char*** (et **jit.scalebias** en particulier), voici quelques exemples supplémentaires.

• Un par un, cliquez sur chacun des presets de l'objet *preset* en procédant de gauche à droite. Dans les paragraphes ci-dessous, nous expliquons chacun des exemples de préréglages.

1. La valeur dans le plan vert de la matrice origianle est 255. (C'est égal à 1,0 sur une échelle de 0 à 1.) L'objet *jit.scalebias* multiplie cette valeur par 0,5, ce qui donne une valeur interne de 127,5; cependant, lors du stockage de la valeur en tant que ***char***, *jit.scalebias* tronque (coupe) la partie fractionnaire et stocke la valeur sous la forme 127.

   Cela donne un résultat plutôt imprécis. (127 sur une échelle de 0 à 255 est égal à 0,498 sur une échelle de 0 à 1, par opposition à la valeur attendue de 0,5). Mais c'est le mieux que nous puissions faire avec des données **char** sur 8 bits. Dans les cas où vous avez besoin d'une plus grande précision, les données ***char*** ne sont pas pour vous. Vous devez utiliser une matrice de données ***long***, ***float32*** ou ***float64*** et utiliser les objets *jit.op* ***@op*** ***** et *jit.op* ***@op*** **+** à la place.
2. La valeur d'origine est 100 et nous la doublons (facteur d'échelle 2.0) pour obtenir le résultat attendu de 200. Il n'y a pas de perte de précision dans ce cas.
3. La valeur d'origine est 100 (0,392). Nous l'échelonnons par un facteur de 1,0, ce qui la laisse inchangé, puis nous lui ajoutons -0,2 (c'est-à-dire que nous en soustrayons 0,2) pour obtenir un résultat de 49 (soit 0,192).
4. 0,392 fois 2,0 plus 0,2 = 0,984. Sur une échelle de 0 à 255, cel correspond à 251.
5. Cet exemple et l'exemple suivant montrent ce qui se passe lorsque le résultat des opérations de multiplication et d'addition dépasse la capacité d'un ***char*** 8 bits. *jit.scalebias* va simplement écrêter (limiter) le résultat à la limite maximale ou

minimale d'un ***char***. Ici, 0,392 fois 4,0 est égal à 1,568 (c'est-à-dire que 100 fois 4 est égal à 400), le résultat est fixé au maximum autorisé, 255.

6. Dans l'autre sens, 0,392 moins 0,5 est égal à -0,108, le résultat est donc fixé à 0.

Il est à noter que ces imprécisions et limites ne se produisent qu'au moment où le résultat est ré-enregistré sous forme de ***char***. Jusque-là, les valeurs sont calculées en interne sous forme de flottants, et la précision est donc conservée. Même si la multiplication fait sortir la valeur interne au-delà de la plage 0-1, aucune limite ne se produit en interne et l'opération d'addition peut la ramener dans la plage. Ici, 0,392 fois 3,0 (= 1,176) moins 0,5 est égal à 0,676. Lorsque cette valeur est stockée sous la forme d'un ***char***, une petite imprécision apparaît. 0,676 sur une échelle de 0 à 255 est égal à 172,38, mais la partie fractionnaire est tronquée et la valeur stockée est 172 (c'est-à-dire 0,675). Pour qu'il n'y ait pas de changement, le facteur d'échelle doit être de 1 et le décalage de biais doit être de 0.

Essayez quelques valeurs vous-même, jusqu'à ce que vous soyez satisfait de votre compréhension de *jit.scalebias* et des résultats qui se produisent avec les données de ***char*** 8 bits. Lorsque vous avez terminé, fermez la fenêtre [***explain_scalebias***].

Ajuster les niveaux de couleur des images

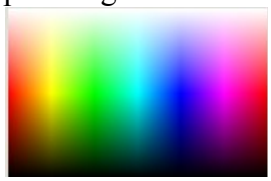
Appliquons maintenant *jit.scalebias* aux images en couleur. Dans le coin supérieur gauche du patch du didacticiel, vous pouvez voir une configuration familière: un objet *jit.movie* avec une boîte de *message* **read** pour charger un film et un objet *metro* pour déclencher les messages **jit_matrix** à partir de *jit.movie*. Dans ce patch, nous allons modifier les matrices avec multiplications et additions dans *jit.scalebias*.

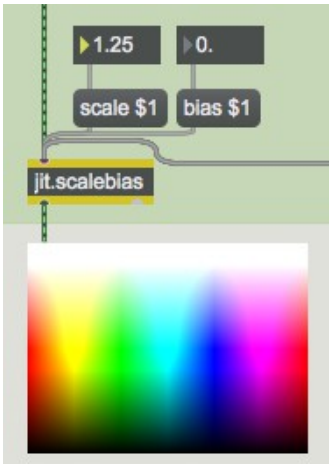
Charger une image ou un film.

• Cliquez sur la boîte de *message* **read chilis.jpg** pour lire une image JPEG. Notez que nous lisons une image fixe - pas une vidéo - dans l'objet *jit.movie*. QuickTime peut gérer une grande variété de formats de médias, y compris les images fixes au format PICT ou JPEG. *jit.movie* traite les images fixes comme s'il s'agissait de vidéos d'une image.

La sortie de *jit.movie* ira à *jit.scalebias* pour traitement, puis sera ensuite affichée dans *jit.pwindow*. (Vous pouvez ignorer l'objet *jit.matrix* pour l'instant. Nous en discuterons plus tard de son utilisation.) Les valeurs ***scale*** et ***bias*** peuvent être changées en modifiant les attributs **scale** et **bias** de *jit.scalebias*.

• Cliquez sur le *toggle* pour démarrer le *metro*. Essayez de faire glisser la boîte de *nombre* située au-dessus de la boîte de *message* **scale $ 1** pour augmenter la valeur de l'attribut ***scale*** à **1.25**.





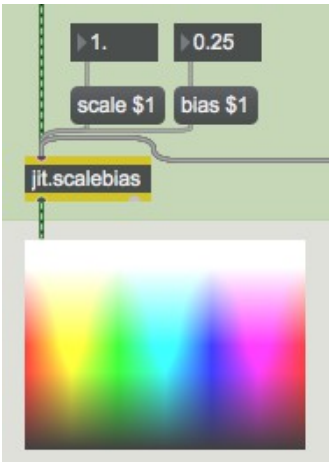
Augmenter la luminosité de l'image; avec ***scale***, plus les valeurs sont élevées, plus elles sont boostées.

Cela augmentera toutes les valeurs non nulles dans les quatre plans de l'image d'un facteur 1,25 (soit une augmentation de 25%). Notez que la multiplication a pour effet d'augmenter les valeurs les plus grandes d'une quantité supérieure aux valeurs les plus petites. Par exemple, si la valeur rouge d'une cellule particulière dans la matrice d'origine est de 200, elle sera portée à 250 (une augmentation nette de 50), tandis que la valeur bleue de la même cellule pourrait être de 30 et serait portée à 37 (une augmentation nette de 7).

• Essayez d'augmenter l'attribut ***scale*** à une valeur très élevée, telle que **20**. Les valeurs qui étaient supérieures ou égales à 13 dans la matrice d'origine seront poussées au maximum de 255 (et même les très petites valeurs seront augmentées à un niveau visible), créant un aspect artificiellement "surexposé".

• Essayez de réduire l'attribut ***scale*** à une valeur comprise entre 0 et 1. Comme on peut s'y attendre, cela assombrit l'image. Une valeur ***scale*** égale ou inférieure à 0 définit toutes les valeurs sur 0.

• Renvoyez l'attribut ***scale*** à **1**. Maintenant, essayez d'ajuster l'attribut ***bias***. Cela ajoute une quantité constante à toutes les valeurs de la matrice. Les valeurs positives éclaircissent l'image et les valeurs négatives l'assombrissent.



Augmentez (ou réduisez) le niveau de toutes les valeurs d'une valeur constante.

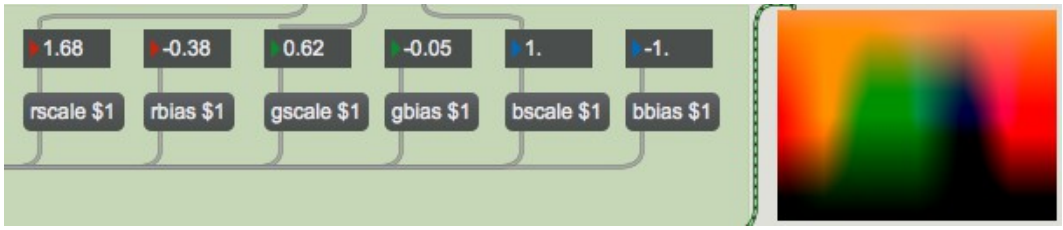
• Voici quelques réglages de ***scale*** et de ***bias*** plus extrêmes que vous pourriez vouloir essayer. Réglez la valeur ***scale*** sur 40 et la valeur ***bias*** sur -20. Cela a pour effet de pousser presque toutes les valeurs à 255 ou à 0, ne laissant que quelques couleurs autres que le blanc ou le noir. Maintenant, essayez de régler la valeur ***scale*** sur -1 et la valeur ***bias*** sur 1. Cela a pour effet d'inverser le schéma de couleurs en faisant en rendant toutes les valeurs élevées faibles et toutes les

valeurs faibles élevées. Réduire encore plus la valeur ***scale*** (par exemple jusqu'à -4 ou -8) crée une inversion similaire, mais seules les valeurs qui étaient faibles dans l'original seront ramenées dans la plage 0-1 par la valeur d'un ***bias*** positive.

Ajustez les plans individuellement

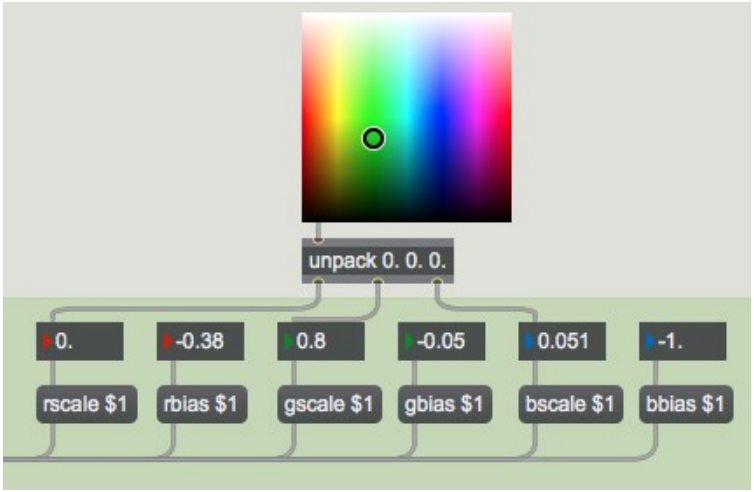
Vous pouvez ajuster les niveaux de chaque plan individuellement dans *jit.scalebias*, en utilisant les attributs **ascale**, **abias**, **rscale**, **rbias**, etc.

• Remettez la valeur ***scale*** à 1 et la valeur ***bias*** à 0. Ensuite, essayez d'ajuster chaque plan de couleur indépendamment en fournissant de nouvelles valeurs pour les attributs appropriés.



Ajustez les niveaux pour chaque plan de couleur

Nous avons rendu le processus un peu plus interactif en vous donnant un contrôleur qui vous permet d'ajuster l'échelle des trois plans de couleur en même temps. Lorsque vous cliquez ou faites glisser l'objet *swatch*, il envoie une liste de trois éléments représentant les valeurs de couleur RVB à l'endroit où se trouve votre souris. Ces valeurs étaient autrefois exprimées sur une échelle de 0 à 255, mais cette échelle a été modifiée pour produire des valeurs de 0,0 à 1,0. (si vous le souhaitez, l'inspecteur contient une case à cocher pour fournir l'ancien style de sortie.) Nous utilisons *unpack* pour diviser cette liste en trois flottants distincts, et nous utilisons ces valeurs pour modifier les attributs **rscale**, **gscale** et **bscale** de *jit.scalebias*.



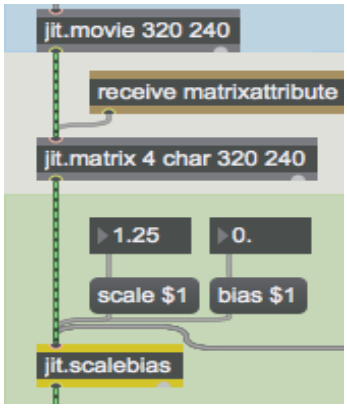
Valeurs de nuance utilisées comme valeurs d'attribut pour *jit.scalebias*

• Faites glisser l'objet *swatch* pour mettre à l'échelle tous les plans RVB en même temps. Comme cette opération produit des valeurs de mise à l'échelle comprises entre 0 et 1, vous réduisez effectivement tous les niveaux. Cela assombrit donc quelque peu l'image résultante.

• Vous pouvez essayer toutes ces opérations sur différentes images. Lisez d'autres images colorées telles que ***colorswatch.pict*** ou ***wheel.mov*** et expérimentez le réglage des niveaux de couleur.

Réaffectation des plans d'une matrice

Dans le tutoriel précédent, nous avons utilisé *jit.unpack* et *jit.pack* pour réassigner les plans d'une matrice. Il existe un autre moyen de le faire, en utilisant l'attribut **planemap** de l'objet *jit.matrix*. Dans cet exemple, nous passons la sortie de la matrice de *jit.movie* à travers un objet *jit.matrix* juste pour illustrer l'attribut **planemap**.



Nous pouvons réaffecter les plans d'une matrice lorsqu'elle passe par *jit.matrix*.

L'attribut **planemap** de *jit.matrix* nous permet de mapper (assigner) n'importe quel plan de la matrice entrante à n'importe quel plan de la matrice sortante. Le mot plan est suivi d'autant de nombres qu'il y a de plans dans la matrice (quatre dans ce cas). Chaque place dans la liste représente un plan de sortie (la première place représente le plan de sortie 0, la deuxième place le plan de sortie 1, etc.) et la valeur de ce nombre indique le plan d'entrée qui lui sera assigné. Par défaut, les valeurs de **planemap** sont 0 1 2 3 (etc.), de sorte que chaque plan de la matrice d'entrée est affecté au même plan de la matrice de sortie. Mais nous pouvons modifier ces affectations à notre guise. Par exemple, si nous envoyons à *jit.matrix* le message **planemap 0 3 2 1**, nous affectons le plan d'entrée 3 au plan de sortie 1 (puisque la valeur 3 est dans la liste de localisation du plan de sortie 1) et le plan d'entrée 1 au plan de sortie 3. En fait, nous échangeons les plans de couleur rouge et bleu de l'image.

• Cliquez sur la boîte de *message* **read wheel.mov** et démarrez le *metro* pour afficher le film. (Définissez l'attribut **scale** de *jit.scalebias* à **1** et l'attribut **bias** à **0** afin d'avoir une image non modifiée dans la fenêtre *jit.pwindow*.) Maintenant, dans la partie inférieure droite du patch, cliquez sur la boîte de *message* **planemap 0 3 2 1** pour permuter les plans rouge et bleu dans la matrice. Cliquez sur la boîte de message **planemap 0 1 2 3** pour revenir au mappage du plan normal. Si nous réglons les trois plans de sortie RVB sur le même plan d'entrée, nous obtenons des valeurs égales dans les trois plans RVB, ce qui donne une image en niveaux de gris.

• Cliquez sur la boîte de *message* **planemap 0 1 1 1** pour voir cet effet. La valeur **1** figure dans l'emplacement de la liste pour chacun des trois plans RVB, de sorte que le plan rouge de l'original est utilisé pour les trois plans RVB de la matrice de sortie. Pour faire pivoter toutes les rotations des différents plan de couleur, nous avons rempli un objet *coll* avec diverses affectations de plan (de la même manière que dans le chapitre précédent) et nous allons envoyer ces affectations à la matrice *jit.matrix* pour modifier les paramètres de son attribut **planemap**.

• Double-cliquez sur l'objet *patcher* **rotatecolorplanes** pour voir le contenu du sub-patch. Il compte simplement de 1 à 6, pour passer en revue les différentes mappings stockés dans l'objet *coll* du patch principal. (Et lorsqu'il est désactivé, il envoie le chiffre **1** pour réinitialiser le mappage de plan par défaut.) Fermez la fenêtre [**rotatecolorplanes**].

• Cliquez sur le *toggle* située au-dessus de l'objet *patcher* **rotatecolorplanes** pour passer en revue les différents mappages de plan à raison d'un réglage par seconde. Modifiez une valeur plus petite (80, par exemple) sur la boîte de *nombre* située au-dessus de l'entrée de droite pour obtenir un effet de scintillement dû à la réaffectation rapide dans le plan.

Dans le chapitre suivant du didacticiel, vous apprendrez comment faire pivoter la teinte de l'image de manière plus subtile, en utilisant *jit.hue*, et vous verrez d'autres façons d'ajuster les niveaux de couleur avec l'objet *jit.brcosa*.

Lecture et importation d'images

Dans ce patch, nous avons chargé trois différents types d'images dans l'objet *jit.movie*: des images fixes PICT et JPEG et un film. Il peut sembler un peu étrange de lire des images fixes dans un objet conçu pour la lecture de films, mais en réalité, QuickTime peut lire de nombreux types de fichiers multimédias, et *jit.movie* sait comment les lire tous. (Vous pouvez même lire un fichier audio AIFF dans *jit.movie*, l'écouter avec des messages **start** et **stop**, sauter à différents endroits avec l'attribut **time**, etc.! Bien entendu, vous ne recevrez aucune information visuelle de la matrice dans ce cas .)

Pour les images fixes, il est tout aussi facile de les charger directement dans un objet *jit.matrix* avec le message ***importmovie***, comme illustré dans le *didacticiel 3*. Si vous importez un film de cette manière, une seule image de celui-ci sera stockée dans le *jit. Matrix*.

Dans ce patch, nous avons utilisé *jit.movie* pour charger toutes les images. La première raison est qu'une des choses que nous voulions charger était un film (pas seulement une image du film). La deuxième raison est que nous voulions démontrer l'attribut **planemap** de *jit.matrix*. L'attribut **planemap** n'est approprié que s'il existe une matrice d'entrée réelle (un message **jit_matrix** arrivant dans l'entrée gauche). Si nous avions importé les images directement dans *jit.matrix* avec ***importmovie***, l'attribut **planemap** n'aurait aucun effet.

Résumé

L'objet *jit.scalebias* utilise la multiplication et l'addition pour modifier toutes les valeurs dans un plan particulier d'une matrice ***char*** à 4 plans, ou tous les plans en même temps. L'attribut **scale** est un facteur par lequel chaque valeur de la matrice sera multipliée. L'attribut **bias** est un montant à ajouter à chaque valeur après que la multiplication ait eu lieu. Les attributs **scale** et **bias** affectent les quatre plans de la matrice. Pour n'affecter qu'un seul plan à la fois, utilisez les attributs correspondant à ce plan, tels que **ascale**, **abias**, **rscale**, **rbias**, etc.

Vous devez fournir les valeurs de ces attributs sous forme de nombres **float**s (nombres contenant un point décimal). Pour effectuer les opérations de multiplication et d'addition, *jit.scalebias* traite les valeurs des ***chars*** comme des valeurs fractionnaires comprises entre 0 et 1, effectue les calculs avec des flottants, puis reconvertit les résultats en ***chars*** (nombres entiers de 0 à 255) lorsqu'il les réenregistre. Les résultats qui dépassent la plage de 0 à 1 seront limités à 0 ou à 1 avant d'être reconvertis en ***chars***.

Vous pouvez réaffecter les plans d'une matrice en utilisant l'attribut **planemap** de *jit.matrix*. Les arguments de **planemap** sont les plans de sortie listés dans l'ordre, et les valeurs de la liste sont les plans d'entrée à affecter à chaque plan de sortie. Ainsi, par exemple, pour affecter le plan 1 d'une matrice d'entrée aux quatre plans de la matrice de sortie, l'attribut doit être défini comme étant **planemap 1 1 1 1**.

jit.scalebias fournit un outil puissant pour ajuster les niveaux de couleur dans une matrice de caractères à 4 plans (couleur ARVB). D'autes outils de ce type sont présentés dans le chapitre suivant du tutoriel.