9-Plus de mixage

Ce tutoriel explore certaines techniques de fondu enchaîné qui peuvent être réalisées avec *jit.scalebias* et *jit.op*. Bien qu'il s'agisse d'une méthode plus complexe que l'approche simple de *jit.xfade*, elle peut offrir plus de flexibilité.

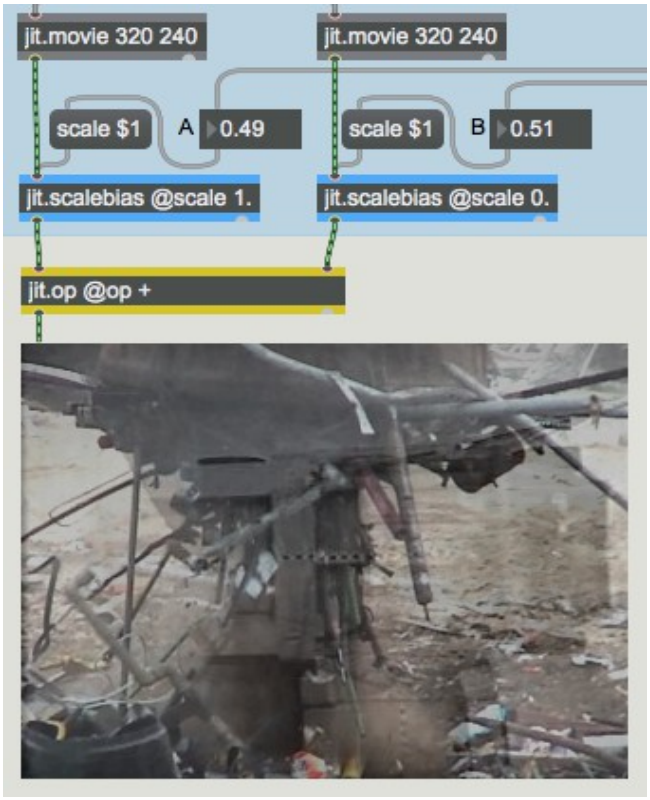
Le mixage et le crossfading sont explicites

Dans le chapitre précédent, nous avons expliqué comment l'objet *jit.xfade* utilise la mise à l'échelle (multiplication) et l'addition pour mélanger deux matrices dans des proportions variables. Dans ce tutoriel, nous allons utiliser les objets *jit.scalebias* et *jit.op* pour effectuer ces opérations mathématiques nous-mêmes.

Cela nous offrira quelques avantages. Premièrement, cela nous permettra de démontrer les opérations mathématiques de mixage et de fondu enchaîné de manière explicite. Deuxièmement, cela nous permettra de montrer comment *jit.op* peut effectuer des opérations mathématiques en utilisant deux matrices comme entrées. (Dans le ***didacticiel 3***, nous avons montré *jit.op* en action avec des valeurs scalaires opérant sur une seule matrice.) Troisièmement, cela nous permettra de spécifier les paramètres d'équilibre (facteurs d'échelle) pour les deux matrices indépendamment, offrant plus de flexibilité que l'objet *jit.xfade*. . Enfin, comme *jit.op* peut implémenter de nombreux types d'opérations mathématiques différentes, nous pouvons essayer d'autres moyens de combinaison des matrices pour obtenir différents effets visuels.

Le mixage revu et corrigé

• Ouvrez le patch du tutoriel.



Multiplication et addition aux matrices de mixage / crossfade (réplication de jit.xfade)

Ici, vous voyez chacune des deux vidéos différentes être réduite (assombrie) par un facteur entre 0 et 1 avec *jit.scalebias*. En-dessous, vous voyez une petite nouveauté dans l'utilisation de *jit.op*: l'entrée dans les deux entrées est une matrice. Lorsque nous faisons cela, *jit.op* exécute l'opération

mathématique spécifiée sur chaque valeur individuellement, en associant chaque valeur de la matrice de gauche à la valeur correspondante de la matrice de droite. Cela nous permet d'additionner toutes les valeurs des deux matrices, en mélangeant efficacement les images.

Le résultat de ces multiplications et de cette addition est comparable à ce que l'objet *jit.xfade* effectue en interne. Vous pouvez le vérifier en utilisant les commandes situées dans la partie supérieure droite du patch - qui sont presque identiques à celles du chapitre précédent - pour effectuer un fondu enchaîné des vidéos.

• Démarrez le *metro* et utilisez le curseur ***Mixer*** pour effectuer un fondu enchaîné de la vidéo A à la vidéo B.

Notez que nous envoyons directement la valeur de fondu enchaîné en tant qu'attribut **scale** pour la vidéo B, tout en utilisant un objet ! **-1** pour redimensionner la vidéo A par 1 moins cette valeur. De cette façon, la somme des deux facteurs d'échelle est toujours égale à 1, comme c'est le cas dans *jit.xfade*.

Combiner des matrices à l'aide d'autres opérateurs

L'addition est peut-être l'opération la plus évidente à effectuer avec deux matrices, mais ce n'est pas la seule possible. En modifiant l'attribut **op** de l'objet *jit.op*, nous pouvons essayer de nombreuses autres opérations pour voir quel type d'effet visuel elles créent.

• Définissez un temps de fondu enchaîné très progressif dans la boîte de *nombre* ***temps de transition*** (par exemple, **10000** ms). Choisissez un opérateur autre que + dans le menu déroulant ***Opérateur***. Maintenant, cliquez sur ***Go To switch*** pour lancer le fondu enchaîné. Vous pouvez voir à quoi ressemble cet opérateur particulier lorsqu'il est mis en oeuvre avec deux matrices vidéo.

Le menu contextuel contient quelques-uns des nombreux opérateurs fournis par *jit.op*. Voici une brève description de chaque opérateur dans le menu.

1. + Ajoutez les valeurs de B à A.
2. **-m** Soustrayez les valeurs de B de A, puis effectuez une opération modulo pour replacer le résultat dans la plage souhaitée.
3. **max** Utilisez la valeur la plus grande, A ou B.
4. **absdiff** Soustrayez les valeurs de B de A, puis utilisez la valeur absolue de cette différence.
5. | "Bitwise or"; en utilisant des valeurs numériques bianires, chaque fois qu'un bit est égal à 1 dans A ou B, il est mis à 1 dans le résultat.
6. ^ "Bitwise Exclusive Or"; en utilisant des valeurs numériques, chaque fois que des bits de A et B ne sont pas identiques, définissez ce bit sur 1 dans le résultat, sinon définissez le bit sur 0.
7. > Si la valeur de A est supérieure à la valeur de B, définissez le résultat sur 1 (ou **char** 255), sinon il prend la valeur 0.
8. **<** Si la valeur de A est inférieure à la valeur de B, définissez le résultat sur 1 (ou **char** 255), sinon mettez-le sur 0.
9. **>** p Si la valeur de A est supérieure à la valeur de B, utilisez la valeur A dans le résultat, sinon mettez-le sur 0.
10. **<** p Si la valeur de A est inférieure à la valeur de B, utilisez la valeur A dans le résultat, sinon mettez-le sur 0.

Si vous voulez voir quels autres opérateurs sont disponibles, consultez la documentation ***Object Reference*** pour *jit.op*.

• Si vous le souhaitez, vous pouvez glisser directement sur les boîtes de *nombres* situées au-dessus des objets *jit.scalebias*, pour définir les niveaux d'équilibre indépendamment (c'est-à-dire différemment de la façon dont notre schéma de fondu enchaîné les définit). Vous pouvez également essayer des valeurs qui dépassent la plage de 0 à 1.

jit.scalebias contre *jit.op @op **

Nous avons choisi d'utiliser l'objet *jit.scalebias* dans ce patch pour effectuer les multiplications d'échelle au lieu d'utiliser *jit.op* avec l'opérateur **. Pourquoi? Lorsque *jit.op* exécute des opérations sur des données **char** (comme nous le faisons dans ce patch), il limite son attribut **val** à la plage 0.-1. (quand il est spécifié comme un float) ou 0-255 (lorsque spécifié comme un int). Dans les cas où nous voulons multiplier les données **char** par une certaine quantité comprise entre 0. et 1., *jit.op* convient parfaitement, car il permet des facteurs d'échelle qui dépassent la plage de 0. à 1. *jit.scalebias* sert uniquement à gérer les matrices **char** à 4 plans, mais c'est OK car c'est ce que nous mettons à l'échelle dans cet exemple. Donc, dans ce patch, puisque nous opérons sur des matrices **char** à 4 plans, et puisque nous voulons que vous ayez la possibilité d'essayer des facteurs d'échelle qui dépassent la plage de 0. à 1., nous avons utilisé *jit.scalebias*.

Sommaire

Vous pouvez utiliser l'objet *jit.op* pour effectuer diverses opérations mathématiques en utilisant des valeurs de deux matrices différentes. *jit.op* effectue l'opération mathématique spécifiée sur chaque valeur individuellement, en appariant chaque valeur de la matrice de gauche avec la valeur correspondante de la matrice de droite. Lorsque les attributs **dim**, **planecount** et **type** des deux matrices diffèrent, *jit.op* utilise les attributs de la matrice dans l'entrée gauche. Différents opérateurs mathématiques peuvent créer une variété d'effets visuels lorsqu'ils sont utilisés pour combiner deux images vidéo.