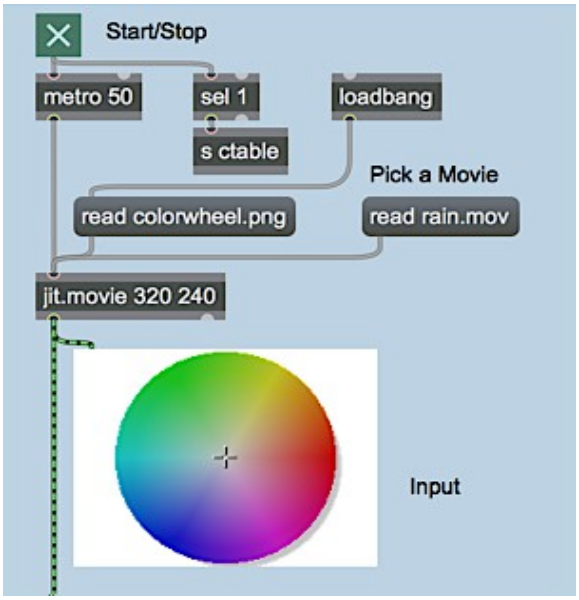
12-Tables de consultation des couleurs

Dans ce didacticiel, nous allons explorer comment utiliser les tables de consultation des couleurs pour remapper des données dans une matrice Jitter. L'objet *jit.charmap* est un outil puissant pour cela. Nous examinerons également différentes stratégies pour générer des matrices à utiliser comme tables de conversion, notamment l'objet *jit.gradient*.

Le processus de consultation

Les tables de consultation (souvent appelées fonctions de transfert) sont des tableaux de nombres où un nombre d'entrée est utilisé comme index dans le tableau. Le numéro stocké à cet index (ou adresse, ou position) est ensuite récupéré pour remplacer le nombre original. Toute fonction – un graphique où chaque valeur x (adresse) a une valeur y correspondante (sortie) - peut être utilisée comme une table de consultation. Les objets max, tels que *funbuff*, *table* et l'objet MSP *buffer~* sont des candidats courants pour l'utilisation de tables de consultation. Dans ce tutoriel, nous utiliserons les matrices Jitter de la même manière.

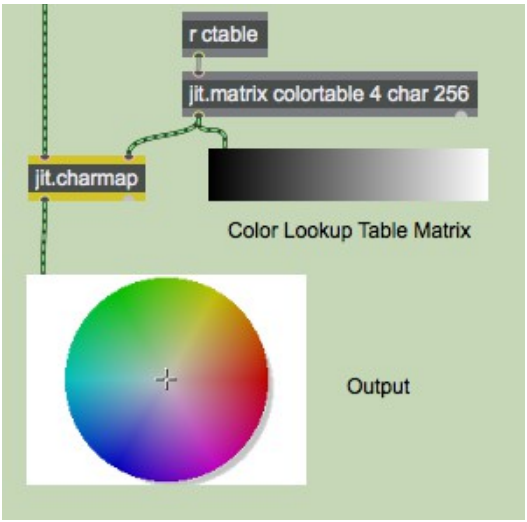
• Ouvrez le patch du tutoriel.



Lire les images

La partie supérieure gauche du patch montre un objet *jit.movie* dans lequel on peut lire deux fichiers différents. L'objet est initialisé (via *loadbang*) avec le fichier ***colorwheel.pct*** chargé dans celui-ci. Vous pouvez également charger le film ***rain.mov*** en cliquant sur la boîte de *message* indiquant **read rain.mov**. Vous pouvez alterner entre ces deux sources d'images tout au long du tutoriel.

• Démarrez le *metro* en cliquant sur le *toggle* en haut du patch. Vous verrez la roue des couleurs apparaître à la fois dans la fenêtre *jit.pwindow* en haut et dans la fenêtre *jit.pwindow* en bas du patch. En outre, vous verrez un gradient apparaître dans une troisième fenêtre *jit.pwindow* (rectangulaire) en bas.



La sortie de *jit.charmap* et la matrice de la table de recherche

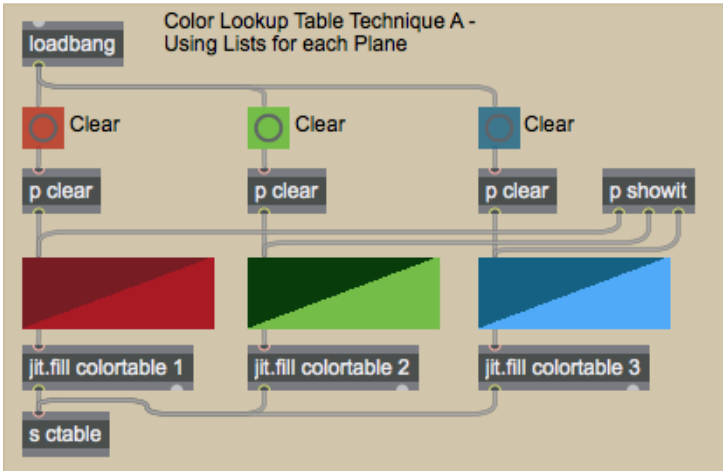
La partie inférieure du patch contient un objet *jit.charmap* que nous utiliserons dans ce didacticiel pour remapper les valeurs de cellules dans l'image. L'objet a deux entrées, dont la gauche est connectée à notre objet *jit.movie* en haut du patch. L'entrée droite est connectée à une matrice *jit.matrix* nommé **colortable**. La matrice **colorable** est unidimensionnelle, avec quatre plans de ***char*** et une largeur de **256** cellules. C'est la table de consultation que *jit.charmap* utilise pour remapper les valeurs de couleur des cellules de la matrice de gauche. L'objet *receive* (en abrégé *r*) avec le nom **ctable** reçoit des données provenant d'ailleurs dans le patch et les envoie à *jit.matrix*. Par exemple, le fait d'activer le *toggle* situé en haut du patch enverra un **bang** à l'objet **colorable** *jit.matrix.* lui faisant envoyer alors son message de matrice à la fois à *jit.pwindow* et à *jit.charmap*.

L'objet *jit.charmap* construit sa matrice de sortie en utilisant les valeurs de la matrice d'entrée (gauche) pour pointer vers des positions dans la matrice (droite) et en copiant la valeur qui s'y trouve. Par exemple, supposons que la matrice que nous envoyons à *jit.charmap* contient une cellule avec les valeurs **100 50 35 20** dans ses quatre plans. Si notre table de correspondance contient la valeur **73** à la cellule **100** du premier plan, **25** dans la cellule **50** du deuxième plan, **0** dans la cellule **35** du troisième plan et **203** dans la cellule **20** du quatrième plan, notre cellule de sortie contiendra le valeurs **73 25 0 203**.

Les tables de recherche pour *jit.charmap* doivent être des matrices unidimensionnelles de **256** cellules avec le même nombre de plans que la matrice que vous souhaitez remapper. En effet, la plage de valeurs possibles pour les matrices **char** est de 0 à 255, donc 256 nombres sont nécessaires pour couvrir la gamme complète de la table de recherche.

Génération de la table de conversion

La partie supérieure droite du patch du didacticiel contient trois objets *multislider* qui vous permettent de concevoir les fonctions de transfert pour les plans 1 à 3 de la matrice **colortable**:



Remplir la matrice de la table de recherche avec les valeurs d'un *multislider*.

Les objets *multislider* (qui ont 256 curseurs entiers dans la plage 0-255) envoient leurs listes aux objets *jit.fill* situés en dessous. Ces objets remplacent les valeurs actuellement stockées dans les plans 1-3 (c'est-à-dire rouge, vert et bleu) de la matrice **colortable** par les valeurs provenant des objets *multislider*. (Voir le ***tutoriel 11***.) Lorsque la matrice a été modifiée avec les nouvelles valeurs, les objets *jit.fil*l envoient un **bang**, que nous envoyons à la matrice *jit.matrix* située à droite du patch qui se connecte à l'entrée droite de *jit. charmap*. Nous ignorons le plan 0 dans ce didacticiel, car il ne contient que des valeurs Alpha lorsque nous traitons les matrices Jitter à 4 plans comme des vidéos.

Les objets *jit.matrix* et *jit.fill* de notre patch partagent le même nom (**colortable**). En conséquence, les deux objets lisent et écrivent dans la même matrice, permettant à un objet (*jit.fill*) de générer des données que l'autre objet (*jit.matrix*) peut lire sans avoir à copier les données entre deux matrices séparées. Ceci est similaire à la façon dont de nombreux objets MSP (par exemple, *peek ~*, *play ~*, *groove ~*) peuvent partager des données d'échantillon stockées dans un seul *buffer ~*. Reportez-vous aux ***didacticiels 11, 16 et 17*** pour plus d'informations sur l'utilisation des matrices nommées.

• Faites quelques dessins à main levée dans les objets *multislider*, pour voir comment cela affecte à la fois la table de consultation (le plus petit des objets *jit.pwindow*) et l'image de sortie de l'objet *jit.movie*. N'oubliez pas de faire des allers-retours entre les deux sources d'images.

Si vous souhaitez réinitialiser l'un des plans à une fonction de transfert y = "x" (c'est-à-dire une ligne droite ascendante qui laisse toutes les valeurs inchangées), vous pouvez cliquer sur l'objet *button* situé au-dessus de l'objet *multislider* concerné. Les sub-patchers appelés *p* **clear** initialisent les objets *multislider* avec un *uzi*.

> Remarque importante: comme de nombreux objets Max, les objets Jitter conservent les matrices stockées dans une entrée même si une nouvelle matrice arrive dans une autre entrée. L'objet *metro* de ce patch n'a donc besoin de déclencher que l'objet *jit.movie*. L'objet *jit.matrix* qui contient la table de consultation de *jit.charmap* n'a besoin de transmettre sa valeur à l'objet que lorsque les données qui y sont stockées changent réellement.

Voici quelques tables de consultation et leurs résultats:

Trois ensembles d'objets *multislider* et leurs tables de recherche de couleurs résultantes et leurs roues de couleurs de sortie

Dans le premier exemple, les fonctions de transfert du rouge et du bleu sont approximativement inversées alors que le vert est normal. Il en résulte que les valeurs élevées de rouge et de bleu dans l'image d'entrée produisent des valeurs faibles sur la sortie, et vice versa. C'est pourquoi le fond blanc de la roue chromatique apparaît maintenant vert (les valeurs de cellule de **0 255 255 255** ont été converties en **0 0 255 0**).

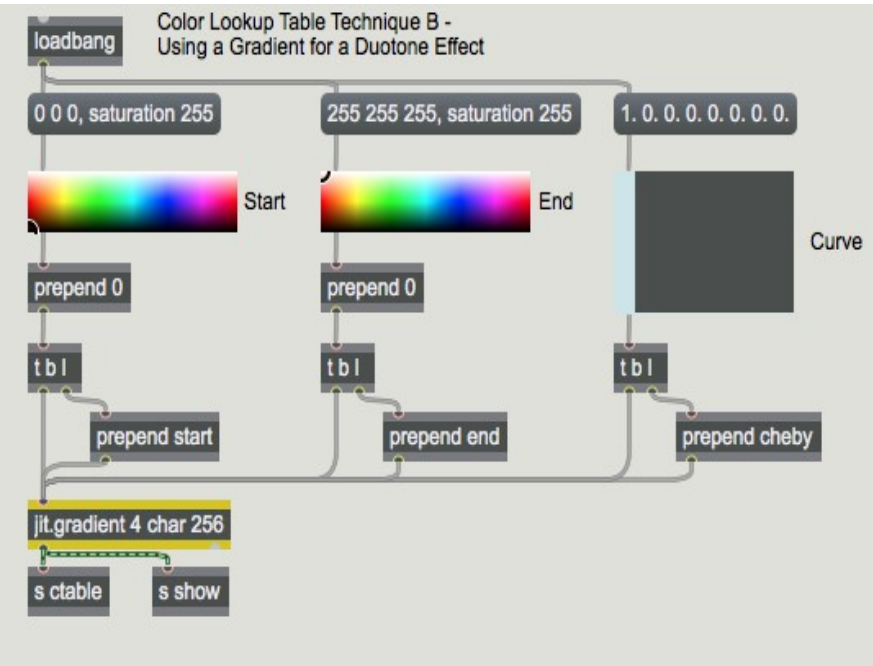
Dans le deuxième exemple, le plan vert est complètement remis à zéro (la fonction de transfert est réglée sur 0 sur toute la plage des valeurs d'entrée). Les plans rouges et bleus sont également mis à 0 jusqu'à un seuil, à partir duquel ils augmentent soudainement (le rouge plus brutalement que le bleu). En conséquence, la majorité de la roue des couleurs est noire (en particulier dans la zone «verte»). Le plan rouge ne devient visible que dans des valeurs très élevées (c'est-à-dire le magenta à l'arrière-plan de la roue chromatique).

Dans le troisième exemple, le plan rouge est représenté normalement. Le plan vert a une forme parabolique, où les valeurs extrêmes sont cartographiées en haut et les nuances moyennes en bas. Le plan bleu est normal, à l'exception d'une plage dans les tons moyens, où il est réduit à zéro. Cette non-linéarité est visible sous la forme d'une «ligne de faille» rouge en haut et en bas du côté droit de la roue des couleurs.

Comme vous pouvez le constater, il existe une infinité de façons de remapper les valeurs des cellules de ces matrices. Nous allons maintenant étudier un autre objet, ce qui nous permet de remapper les valeurs des couleurs de manière plus précise.

L'objet jit.gradient

• Ouvrir le sub-patch **Duotone**.
Cet sub-patch montre une méthode pour générer des tables de consultation à l'aide de l'objet *jit.gradient*:

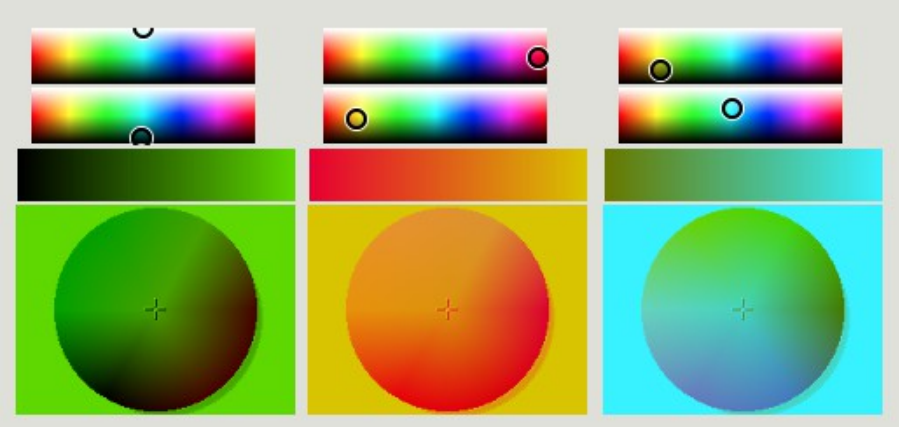


Utilisation de l'objet *jit.gradient*.

L'objet *jit.gradient* génère des matrices ***char*** à une seule dimension qui font une transition douce entre deux valeurs de cellule spécifiées. Les attributs de début et de fin sont des listes qui spécifient ces valeurs de cellule. Par exemple, un attribut **start** de **0 0 0 0** et un attribut **end** de **0 0,5 1,0 0,5** générera un dégradé qui va du noir (à la cellule **0** dans la matrice) au vert pâle (à la dernière cellule de la matrice). Nous avons donné à notre objet *jit.gradient* les arguments appropriés pour qu'il ait une largeur de **256** cellules, de sorte qu'il puisse être stocké dans notre matrice *jit.matrix* lorsqu'il change. Notez que *jit.gradient* utilise des nombres à virgule flottante dans ses listes d'attributs pour spécifier des valeurs ***char*** (c'est-à-dire qu'une valeur de **1,0** dans l'attribut spécifie une valeur ***char*** de **255**).

Les attributs sont formatés en prenant la sortie de liste RVB des objets Max *swatch* et en les convertissant en flottants ARVB. Une fois l'attribut envoyé à l'objet *jit.gradient*, l'objet reçoit un **bang** de l'objet *trigger* qui lui permet de sortir sa matrice dans l'objet *jit.matrix* situé à gauche du patch.

• Essayez de sélectionner quelques couleurs dans les objets *swatch*. Les attributs **start** et **end** spécifieront les limites de la table de conversion, de sorte que les valeurs de l'image d'entrée auront un aspect bicolore, avec un morphing entre ces deux couleurs. Les objets *multislider* situés en haut du patch refléteront les tables de conversion correctes générées par l'objet *jit.gradient*.

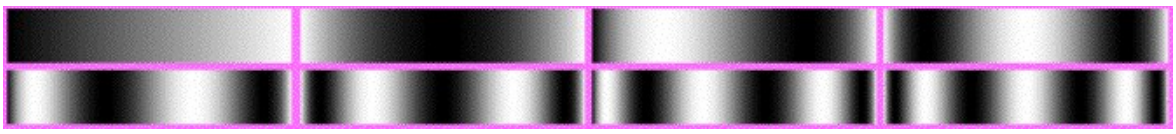


Utilisation de dégradés de couleurs comme tables de consultation

Le premier exemple montre une image inversée. Le début de la table de consultation est **white (start 0 1.0 1.0 1.0)** et la fin de la table de conversion est **black (end 0. 0. 0. 0.)**. En conséquence, les valeurs d'entrée de **0** sont mises en correspondance avec **255**, et inversement (y = 255-x).

Les deuxième et troisième exemples montrent des dégradés bicolores qui remappent le spectre de la roue chromatique entre le rouge et l'orange (exemple 2) et l'olive et le cyan (exemple 3). Remarquez comment, en fonction des couleurs d'origine à différents points de la roue chromatique, la courbe de dégradé devient plus raide ou plus graduelle.

Le troisième attribut de l'objet *jit.gradient* est l'attribut **cheby**, qui spécifie une courbe à suivre lors du morphing entre les valeurs **start** et **end** de la matrice. L'attribut **cheby** prend une liste de nombres à virgule flottante comme arguments. Ces arguments sont les amplitudes de différents ordres de ***polynômes de Chebyshev*** (voir ci-dessous). Ces courbes à fonction spéciale créent des effets différents lorsqu'elles sont utilisées dans des tables de consultation. Le *multislider* du patch du didacticiel qui définit l'attribut **cheby** vous permet de spécifier l'amplitude relative des huit premières courbes des polynomes de Chebyshev, qui ont les formes suivantes (si vous les considérez comme progressant du noir au blanc):

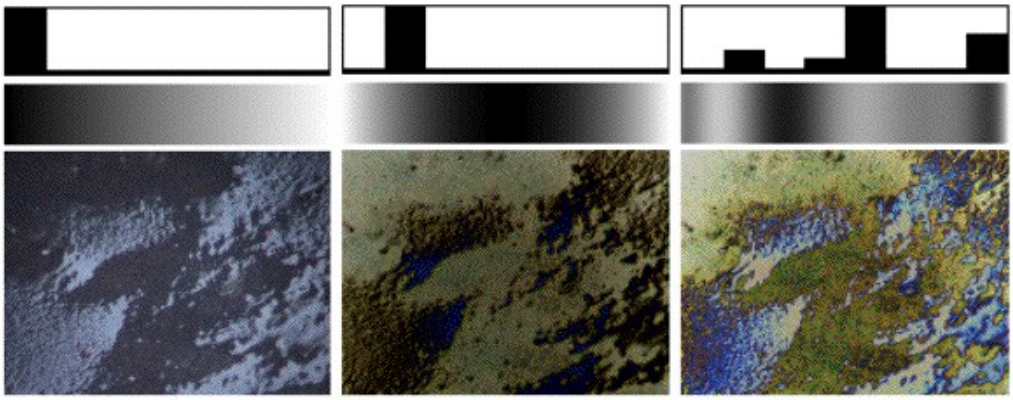


Gradients générés à l'aide des commandes Chebyshev 1-4 (rangée supérieure) et 5-8 (rangée inférieure)

> Note technique: Les polynômes de Chebyshev sont couramment utilisés comme fonctions de transfert dans la mise en forme des signaux audio dans les algorithmes de synthèse numérique (ils ont des propriétés spéciales qui leur permettent de déformer des formes d'ondes sinusoïdales en spectres harmoniques équivalents aux amplitudes de différents ordres). L'objet MSP *lookup* ~ peut être utilisé avec une fonction chargée dans un *buffer* ~ pour effectuer le processus équivalent dans le traitement du signal audio que nous effectuons dans ce didacticiel avec l'image. Voir ***Tutoriel 12: Synthèse: Waveshaping*** dans le manuel MSP pour plus de détails.

• Réinitialisez les points de **start** et **end** du dégradé (en cliquant sur les boîtes de *message* situées comment la roue chromatique change à mesure, les couleurs disparaissant et réapparaissant dans différentes régions.

Lorsque vous utilisez l'attribut **cheby** dans l'objet *jit.gradient*, vous pouvez obtenir des effets de distorsion des couleurs très intéressants, même si vous laissez les points de **start** et **end** du dégradé en noir et blanc. Voici quelques exemples avec notre clip ***rain.mov***:



L'effet de différentes courbes de dégradé sur le spectre de couleurs de la pluie

L'image de gauche montre une image fixe non traitée du film Rain. L'image du milieu montre ce qu'il advient du spectre des couleurs lorsque le dégradé est généré à l'aide d'un polynôme de Chebyshev du second ordre (la zone la plus sombre de l'image se trouve maintenant au centre du spectre des couleurs). L'image de droite montre un dégradé plus complexe, où le spectre de couleurs montre de nombreux pics et creux.

• Les objets *multislider* situés en haut du patch reflètent l'état actuel de notre table de consultation (la matrice générée par notre objet *jit.gradien*t est envoyée à un objet *jit.iter* à l'intérieur du sub-patch *p* **showit**, où les nombres sont groupés pour définir la état des objets *multislider*). Essayez de générer un dégradé puis de modifier manuellement la table de consultation en modifiant les objets *multislider*. Cela vous permet d'utiliser l'objet *jit.gradient* comme point de départ pour une table de conversion plus complexe.

Sommaire

Vous pouvez mapper les valeurs des cellules des matrices **char** *Jitter* en utilisant l'objet *jit.charmap*. L'entrée droite de *jit.charmap* utilise une matrice de **256** cellules qui définit la table de consultation (ou la fonction de transfert) à appliquer aux données de la matrice entrante. Vous pouvez définir la table de consultation à l'aide de plusieurs stratégies, notamment en utilisant *jit.fill* pour générer la matrice à partir de listes Max, ou en utilisant l'objet *jit.gradient* pour générer des dégradés de couleur entre une valeur de cellule **start** et **end** selon une forme de courbe spécifiée par l'attribut **cheby**.