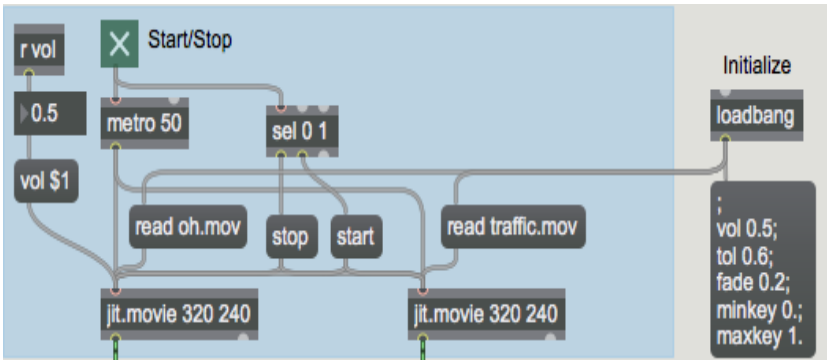
10-chromakeying

Ce tutoriel explique comment effectuer un chromakeying avec deux films sources à l'aide de l'objet *jit.chromakey*. Nous allons également apprendre comment trouver la couleur de n'importe quel pixel de l'écran avec l'objet *suckah*.

Lorsque vous ouvrez le patch du didacticiel, Max lit automatiquement deux films (***oh.mov*** et ***traffic.mov***) dans deux objets *jit.movie* en envoyant des messages **read** appropriés à ces objets avec un *loadbang*:

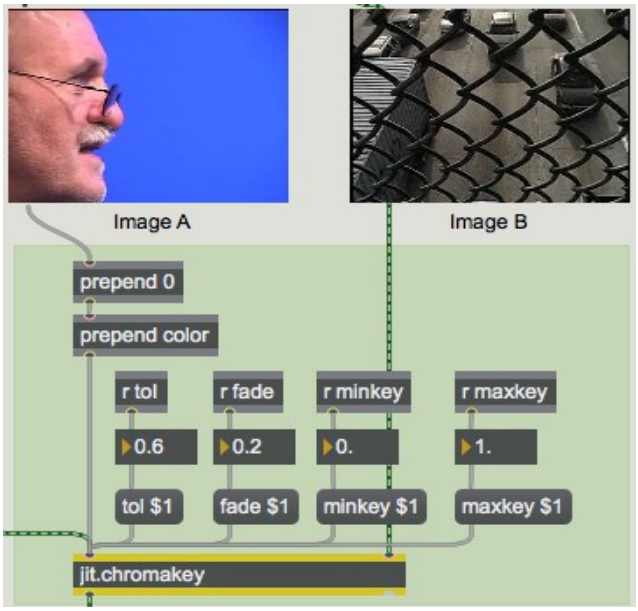


Initialisation du patch via loadbang.

Les paramètres supplémentaires dont nous avons besoin pour ce patch sont également initialisés par le *loadbang*, qui est connecté à la boîte de *messages* située sur la droite du patch. La boîte de *messages* initialise le reste du patch en envoyant des messages à des objets *receive* nommés ailleurs dans le patch. (Voir Didacticiel 16: *Messagerie à distance* - Envoi de messages sans cordons de connexion).

• Cliquez sur le *toggle* pour démarrer *metro*. Vous devriez voir des images apparaître dans les trois objets *jit.pwindow* du patch. Notez que la boîte *toggle* permet non seulement de démarrer et d'arrêter le *metro*, mais aussi de démarrer et d'arrêter le transport du film des deux objets *jit.movie*.

Le troisième objet *jit.pwindow* (dans la partie inférieure droite du patch) ressemble à ceci:



Comment diable est-il arrivé devant cette clôture?

• Cliquez avec la souris sur la région bleue de l'objet jit.pwindow de gauche (c'est-à-dire la zone derrière la tête de l'homme dans le film).

Note historique: Le compositing sur écran bleu, ou le procédé consistant à filmer des séquences en direct sur un fond bleu mat pour remplacer le bleu par une image distincte existe depuis la fin des années 1930. À l'origine, il s'agissait d'un procédé photographique très coûteux impliquant la séparation lithographique des couleurs. L'écran bleu (et son petit frère, l'écran vert) est devenu l'effet spécial le plus courant au cinéma, à la télévision et en vidéo. La possibilité de réaliser le chromakeying (le terme technique pour ce procédé) par superposition numérique n'a fait que le rendre plus omniprésent. Dans le secteur de la télévision, le chromakeying vidéo est souvent appelé CSO (Color Separation Overlay), nom donné au procédé par l'équipe de la BBC qui l'a mis au point dans les années 1960. Petro Vlahos, l'un des innovateurs de l'écran bleu dans les années 1960, s'est vu décerner en 1994 le prix Lifetime Achievement Award de l'Academy of Motion Picture Arts and Sciences, reconnaissant à quel point la technologie était devenue indispensable.

L'objet *jit.chromakey*

L e *Chromakeying* - le processus de superposition d'une image sur une autre par remplacement sélectif de la couleur - est accompli dans Jitter par l'objet *jit.chromakey*. En spécifiant une couleur et quelques autres paramètres, *jit.chromakey* détecte les cellules contenant cette couleur dans la première matrice (à gauche) et les remplace par les cellules équivalentes dans la seconde matrice (à droite) lorsqu'il construit la matrice de sortie. Le résultat est que les cellules de la première matrice se superposent à la seconde.

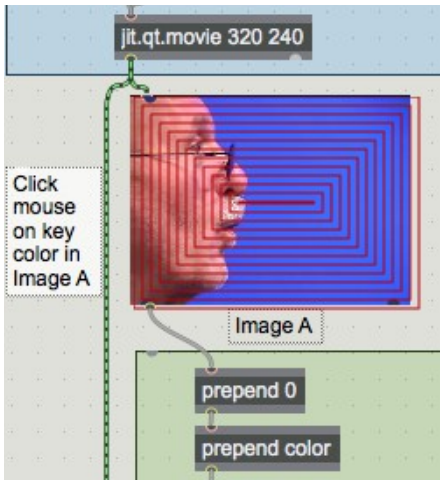
• Puisque n'importe quelle couleur peut être utilisée pour le chromakey, essayez de cliquer ailleurs dans la fenêtre *jit.pwindow* de gauche. Différentes couleurs seront éliminées du visage de l'homme pour révéler le trafic.

Le truc du visage qui disparaît (première partie)

L'objet *jit.chromakey* utilise l'attribut **color** pour définir la couleur centrale dans la chromakey (appelée couleur de référence). Cet attribut est défini comme une liste de valeurs pour autant de plans qu'il en existe dans les matrices qui sont incrustées. L'attribut **tol** spécifie une plage de valeurs autour de la couleur de référence. Les couleurs de cette plage seront également incrustées. Lors de l'utilisation de *jit.chromakey* avec des matrices ***char*** (par exemple,vidéo), les attributs sont spécifiés dans une plage de virgule flottante de **0** à **1**, qui est ensuite mise en correspondance avec la plage **0-255** nécessaire pour les données de ***char***. Par conséquent, pour définir l'attribut **color** d'un chromakey vert uni, vous devez définir l'attribut comme suit : **color 0 0 1.0 0**, et non **0 0 255 0**. Une plage **tol** de **0,5** permet d'enregistrer toutes les valeurs situées à la moitié de la distance chromatique de la couleur de référence (calculée comme la somme des différences absolues entre la couleur de référence et la valeur de cellule réelle dans chaque plan). Une plage **tol** de valeur **0** ne traitera que la couleur de référence exacte dans le cadre de la chromakey.

• Essayez à nouveau de cliquer sur la région bleue dans le film de gauche et jouez avec l'attribut **tol** pour voir comment la sortie de chromakey change. Avec une faible tolérance, une partie de l'écran bleu dans l'image de gauche restera dans la sortie incrustée. Avec une tolérance très élevée, des parties du visage de l'homme peuvent disparaître.

Dans le patch du didacticiel, l'attribut **color** de *jit.chromakey* est défini en cliquant sur un objet invisible. Si vous déverrouillez le patch, vous verrez une région de carrés rouges concentriques qui se trouvent au-dessus de l'objet *jit.pwindow* de gauche:



L'objet *suckah*

La région est un objet d'interface utilisateur Max appelé *suckah*, qui apparaît dans la palette add d'objets : interface comme ceci:

L'objet *suckah* dans la palette d'objets

L'objet *suckah* rapportera les valeurs RVB de tout pixel de l'écran qu'il recouvre. Il rapporte ces valeurs sous forme d'une liste de flottants dans la plage de **0,0** à **1,0** lorsque vous cliquez sur l'objet dans une zone verrouillé. Par exemple, si vous cliquez sur une région de bleu uni sur laquelle se trouve un objet *suckah*, ce dernier enverra la liste **0 0 1.0**. (La première version de *suckah* utilisait la plage **0** à **255** pour la sortie. L'inspecteur contient une case à cocher si ce comportement est

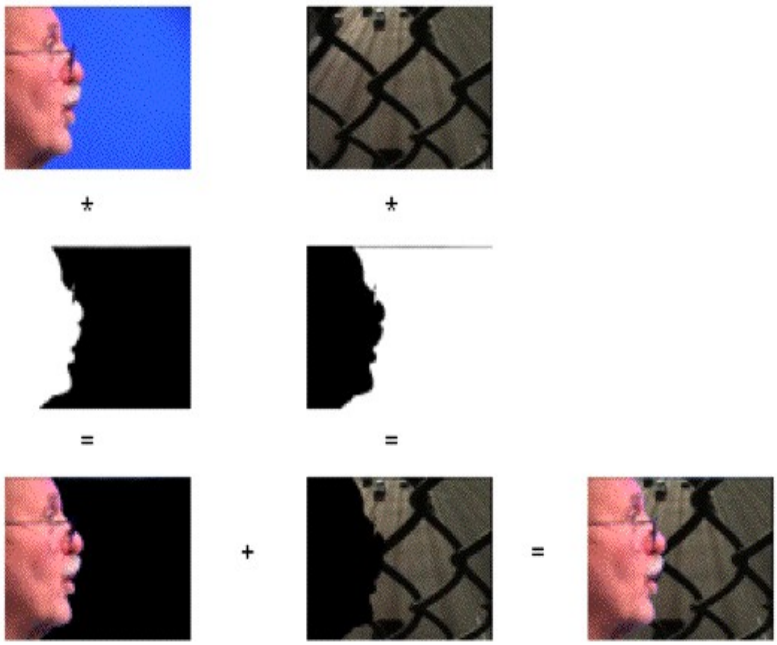
souhaité.)

Pour définir l'attribut **color** de notre objet *jit.chromakey*, nous prenons la liste RVB qui sort de l'objet *suckah* et l'envoyons à travers un *prepend* **0**, qui ajoute une valeur alpha de **0** au début de la liste. Le message est ensuite complété par le *prepend color* et envoyé à *jit.chromakey*.

Options de saisie

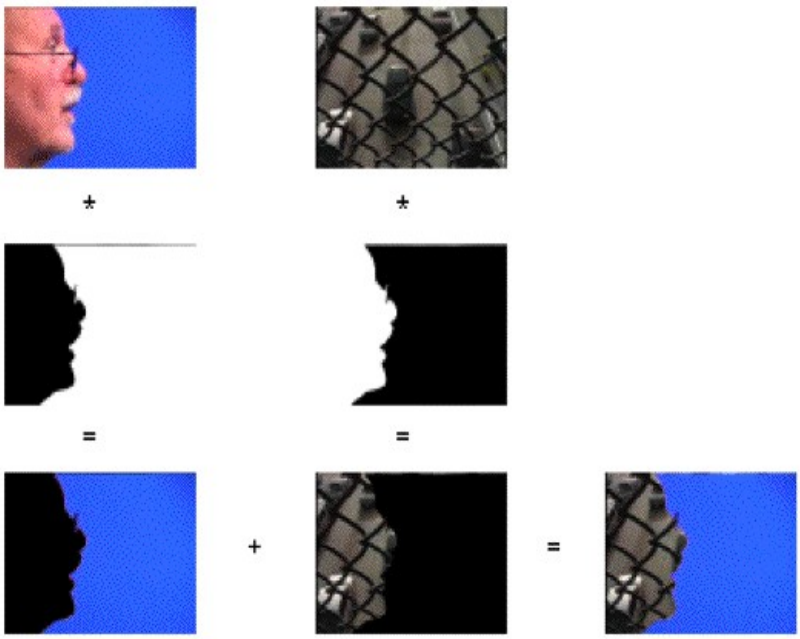
L'objet *jit.chromakey* possède des attributs supplémentaires: **minkey**, **maxkey** et **fade**. Quand une matrice arrive dans l'entrée gauche, *jit.chromakey* crée en interne un masque en niveaux de gris (1-plan) basé sur cette matrice. Les cellules de la matrice entrante qui ont des valeurs de couleur dans la plage de tolérance (**tol**) sont définies à la valeur de l'attribut **maxkey** (la valeur par défaut est **1**) dans le masque. Les régions situées en dehors de la plage de tolérance sont multipliées par l'attribut **minkey** (la valeur par défaut est **0**). Si les attributs **minkey** et **maxkey** sont définis sur **0** et **1**, l'image résultante doit être blanche là où l'incrustation doit avoir lieu, et noire là où l'image originale doit être conservée.

Le masque résultant et son inverse sont ensuite multipliés par les matrices de droite et de gauche, respectivement. Les résultats de la multiplication sont ensuite additionnés pour former l'image composite. Le diagramme suivant vous montre un aperçu imagé du processus:



Les deux sources, leurs masques (avec **minkey** à **0** et **maxkey** à **1**) et *chromakey* composite.

Comme vous pouvez le constater, l'attribut **maxkey** définit la force de la matrice de droite dans la sortie, tandis que l'attribut **minkey** définit la force de la matrice de gauche. Si nous devions inverser les attributs **minkey** et **maxkey**, le *chromakey* serait inversé et les événements suivants se produiraient:



L'effet composite avec **minkey** à **1** et **maxkey** à **0** (touche de *chromakey* inverse).

L'attribut **fade** permet une certaine interpolation entre la zone en cours d'incrustation et la zone non incrustée. Cela vous permet de créer un bord doux pour l'effet *chromakey*. Les couleurs de la matrice de gauche qui sont légèrement en dehors des limites de la plage de tolérance de l'incrustation, mais qui se trouvent dans la plage de **tol** + **fade** de la couleur de référence, sont interpolées entre leur couleur d'origine (non incrustée) et la couleur de la même cellule de la matrice de droite. La quantité d'interpolation dépend de l'importance de la valeur de **fade** et de la distance entre la couleur en question et la plage de tolérance.

• Essayez d'expérimenter avec différentes valeurs de **tol**, **fade**, **minkey**, **maxkey** et **color**. Observez comment les cinq attributs interagissent pour obtenir différents effets d'incrustation et comment les valeurs **minkey** et **maxkey** se complètent.

Le chromakeing précis peut être un processus difficile. Des valeurs correctes pour les attributs **tol** et **fade** sont essentielles pour s'assurer que les bonnes régions de la première image sont incrustées dans la seconde. En général, les images clés très détaillées présentent un léger mouchetage aux endroits où les couleurs se déplacent rapidement entre les régions clés et les régions non clés. De plus, une seule couleur clé (par exemple, le bleu) ne suffit presque jamais pour obtenir une clé complète, il faut toujours utiliser une gamme de valeurs. Vous constaterez souvent, cependant, que la couleur que vous souhaitez éliminer d'une partie de l'image est quelque peu présente dans la région que vous souhaitez conserver! Equilibrer tous ces facteurs pour obtenir l'effet le plus convaincant est la partie la plus difficile de l'utilisation de l'objet *jit.chromakey.*

Sommaire

L'objet *jit.chromakey* vous permet d'effectuer du chromakeying à deux sources dans Jitter. Vous pouvez définir une gamme de couleurs pour la clé en utilisant les attributs **color** et **tol**, et utiliser les valeurs **fade**, **minkey** et **maxkey** pour définir comment les deux matrices fonctionnent dans un composite. L'objet d'interface utilisateur *suckah* vous permet de sélectionner facilement les couleurs telles qu'elles apparaissent à l'écran en définissant l'objet sur une fenêtre *jit.pwindow*. En cliquant sur l'objet *suckah*, vous obtiendrez la couleur du pixel sur lequel vous venez de cliquer à l'écran.