15-Rotation d'image

Rotation et zoom avec jit.rota

Jitter offre un moyen simple de faire pivoter et / ou d'agrandir une image avec un objet appelé *jit.rota*. La rotation et le zoom sont des effets vidéo courants et utiles. En les combinant de différentes manières, vous pouvez également obtenir une variété d'effets kaléidoscopiques. *jit.rota* prend une matrice de données vidéo (ou tout autre type d'image) dans son entrée et envoie une version qui a été agrandie, pivotée ou autrement déformée en fonction des paramètres des attributs de l'objet.

• Ouvrez le patch du tutoriel. La vidéo QuickTime ***dishes.mov*** est lue automatiquement dans l'objet *jit.movie* par un **bang** de *loadbang*. Pour voir la vidéo, cliquez sur le *toggle* d'affichage pour démarrer le *metro*.
La vidéo est un panoramique de la caméra de trois secondes de gauche à droite sur un ensemble de plats. Cependant, l'attribut **loop** de l'objet *jit.movie* ayant été initialisé à **2**, l'animation se répète en boucle, ce qui donne l'illusion d'un panoramique en va-et-vient.

> Remarque: De nombreux attributs des objets Jitter utilisent uniquement les arguments 1 et 0 pour signifier "on" et "off". Il est donc raisonnable de supposer que l'attribut **loop** de *jit.movie* est identique. S'il est vrai que **loop 0** désactive la boucle et **loop 1** l'active, **loop 2** provoque la lecture en avant de la vidéo, puis la lecture en arrière lorsqu'elle atteint le point de **loopend**, plutôt que de sauter au point de **loopstart**.

L'attribut **theta** de *jit.rota* détermine l'angle de rotation autour d'un point d'ancrage central.

• Faites glisser la boîte de *nombre* ***d'angle de rotation*** pour faire pivoter la vidéo. Les valeurs positives (ou croissantes) entraînent une rotation dans le sens inverse des aiguilles d'une montre, et les valeurs négatives (ou décroissantes) entraînent une rotation dans le sens des aiguilles d'une montre. L'angle de rotation - a.k.a. l'angle (***theta***) est indiqué en radians. Une valeur de 0, ou un multiple quelconque de 2π (c'est-à-dire, 6.283185), correspond au positionnement vertical normal. Une valeur de π (c'est-à-dire 3.141593), ou tout multiple impair de π, correspond à la position totalement à l'envers. Expérimentez jusqu'à ce que vous compreniez la relation entre les valeurs **theta** et le comportement de *jit.rota*.

> Détail technique: *jit.rota* effectue de nombreux calculs internes à l'aide de la trigonométrie afin de déterminer la rotation de l'image. Si vous n'êtes pas un amateur de trigonométrie, vous n'êtes peut-être pas habitué à penser aux angles en termes de radians. Dans les conversations de tous les jours, nous utilisons plus souvent des degrés, la rotation complète étant de 360°. En trigonométrie, il est plus courant d'utiliser des ***radians***, où une rotation complète est égale à 2π radians. En effet, un cercle de rayon 1 a une circonférence d'exactement 2π. Vous pouvez donc faire référence à un angle en référençant le point où il intersecterait le cercle unitaire. (Par exemple, si vous avez commencé à un point du cercle unitaire et que vous avez parcouru une distance exactement égale à π/2 autour de la circonférence, vous vous retrouveriez avec un angle de 90°, c'est-à-dire un angle de π/2 radians, à partir de votre point de départ, en référence au centre du cercle.)
> De plus, en trigonométrie, nous considérons un changement d'angle positif comme une rotation dans le sens inverse des aiguilles d'une montre autour du cercle unitaire, alors que dans la vie de tous les jours, un mouvement dans le sens des aiguilles d'une montre peut être plus intuitivement "positif" ou "croissant" (comme le passage du temps).
> Ainsi, pour convertir une rotation en degrés dans le sens des aiguilles d'une montre en une même rotation en radians, vous devez multiplier le degré de l'angle par 2π, puis le diviser

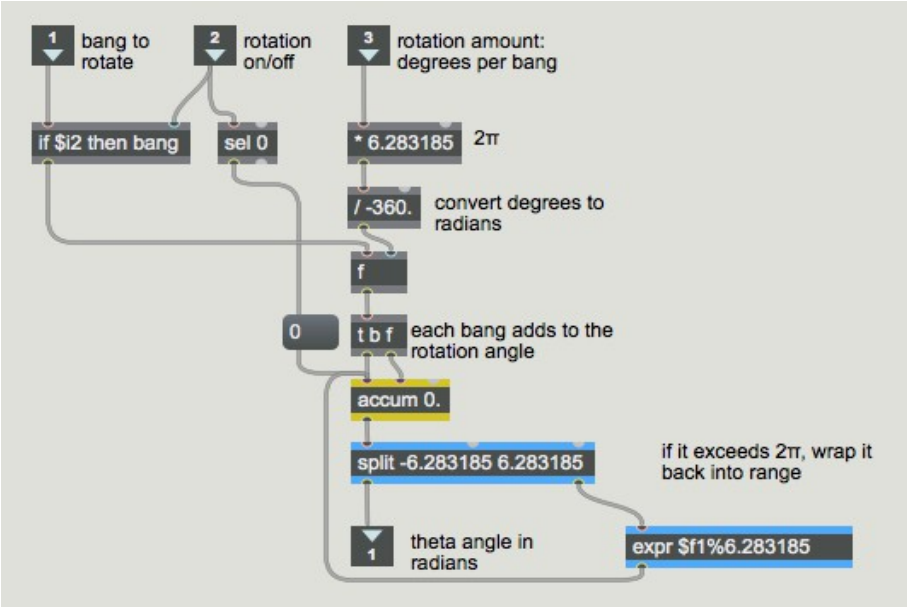
par -360.

Rotation automatisée

Outre la rotation manuelle de l'image, vous pouvez également écrire un processus automatisé dans Max qui fournira des angles de rotation en constante évolution. Dans le chapitre précédent, nous avons écrit un sub-patch appelé ***rotate*** qui utilisait l'objet *line* pour augmenter continuellement l'angle de rotation de la teinte de 0° à 360°. Dans ce chapitre, nous faisons quelque chose de similaire, mais cette fois-ci, nous utilisons le **bang** du *metro* qui affiche le film pour augmenter l'angle de rotation. Pour que cela reste "convivial", nous montrons les ***degrés*** de rotation d'angle de l'utilisateur plutôt que les radians (nous convertissons les degrés en radians à l'intérieur du sub-patch), et nous affichons également la vitesse de rotation sous la forme "rotations par seconde".

• Dans la boîte de *nombre* intitulée ***Degrees per bang***, entrez le chiffre **6**. Cela entraînera une augmentation de l'angle de rotation de 6 degrés à chaque **bang** de *metro*. Étant donné que *metro* envoie 20 bps par seconde (une fois toutes les 50 ms), nous savons que nous pouvons calculer le nombre de rotations par seconde à l'aide de la formule *d*20/360*, c'est-à-dire *d/18*, où d est le degrés d'augmentation d'angle par **bang**. Cliquez maintenant sur le *toggle* marqué ***On***/***Off*** pour commencer la rotation automatisée.

• Double-cliquez sur le patcher de l'objet ***rotate*** pour voir le contenu du sub-patch.



Rotation automatisée dans le sub-patch [rotate]

Nous convertissons ce que l'utilisateur spécifie en degrés par **bang** en un montant en radians, en multipliant les degrés par 2π et en le divisant par –360. (Voir l'encadré Détail technique ci-dessus.) Lorsqu'un **bang** arrive dans l'entrée gauche, si la rotation est activée, le **bang** est passé à travers et provoque une augmentation de la rotation angulaire dans l'objet *accum*. Notez qu'une valeur négative en degrés par **bang** fonctionne également et provoque une rotation de l'image dans le sens anti-horaire. Lorsque l'angle de rotation total dépasse 2π (ou -2π), un *split* envoie la valeur à un *expr* qui utilise une opération modulo pour le ramener dans la plage (réinitialisation de la valeur dans l'objet *accum*) avant de l'envoyer à la sortie. Lorsque la rotation est désactivée, nous détectons ce fait avec un objet *sel* **0** et réinitialisons l'angle ***theta*** à **0**.

• Fermer la fenêtre de sub-patch. Cliquez sur le *toggle* On/Off pour arrêter la rotation automatisée.

Zoom avant ou arrière

L'autre caractéristique principale de *jit.rota* est sa capacité de zoom. La quantité de zoom est déterminée par les attributs **zoom_x** et **zoom_y** de *jit.rota*. Celles-ci vous permettent d'effectuer un zoom avant ou arrière dans les dimensions horizontale et verticale indépendamment; ou vous pouvez zoomer simultanément sur les deux dimensions en modifiant les deux attributs en même temps.

• Faites glisser la boîte de *nombre* intitulée ***Zoom*** pour effectuer un zoom avant ou arrière. Les valeurs supérieures à 1 augmentent l'image (zoom avant) et les valeurs inférieures à 1 réduisent l'image (zoom arrière). Vous pouvez modifier le zoom des dimensions x et y indépendamment en entrant des valeurs directement dans les boîtes de *nombre* x et y. (Les valeurs de zoom négatives retournent l'image et la redimensionnent.)

Lorsque nous effectuons un zoom avant sur l'image, par exemple avec une valeur de zoom de 2, nous conservons toujours une qualité d'image relativement bonne, car nous avons activé l'attribut **interp** de *jit.rota* avec un message **interp 1**. Si vous désactivez l'**interp**, vous obtiendrez une pixellisation lorsque vous effectuerez un zoom avant. Lorsque vous effectuerez un zoom arrière, l'**interp** n'aura aucun effet appréciable, ce qui représente une perte de temps considérable pour l'ordinateur. (Voir le ***tutoriel Jitter 14*** pour une explication de la pixellisation et de l'interpolation.) Cependant, l'interpolation améliore l'apparence des images pivotées, même lorsqu'elles ont été réduites par un zoom arrière.

Au-delà du bord

• Définissez le zoom des deux dimensions sur une valeur faible, telle que **0,25**.

Lorsque l'image ne remplit pas toute la zone d'affichage en raison d'un rétrécissement ou d'une rotation, *jit.rota* doit décider quoi faire du reste de la matrice située en dehors de la zone d'image. Actuellement, *jit.rota* définit toutes les valeurs de cellule en dehors de la zone d'image sur **0**, ce qui les rend toutes noires. La manière dont *jit.rota* traite les cellules situées en dehors des limites de l'image est déterminée par son attribut **boundmode**. Les différents paramètres liés à **boundmode** disponibles sont présentés dans le menu contextuel intitulé ***Space outside the image*** dans le coin supérieur droit du patch. Nous avons initialisé la valeur **boundmode** à **1**, ce qui indique à *jit.rota* de vider toutes les cellules périphériques. Voici la signification de chacun des paramètres liés:

0 ***Ignore***: Laissez toutes les cellules excentrées inchangées par rapport à leurs valeurs précédentes.

1 ***Clear***: Définissez toutes les valeurs de cellules périphériques sur 0.

2 ***Wrap***: Recommencez l'image autant de fois que nécessaire pour remplir la matrice.

3 ***Clip***: pour toutes les cellules périphériques, continuez à utiliser les valeurs des cellules limites de l'image.

4 ***Fold***: Répétez l'image, retournée dans la direction opposée.

• Pour les effets spéciaux lorsque l'image est zoomée, essayez de définir l'attribut **boundmode** sur **2** (wrap) pour un effet d'image dupliqué style "Warhol" ou sur 4 (fold) pour un effet kaléidoscope.

• Essayez maintenant de réactiver la rotation automatisée pour combiner rotation et zoom et modifier les différents paramètres (***Degrees per bang***, ***Zoom*** et ***Space outside the image***).

• Une fois les tests terminés, désactivez la rotation automatique et réglez les attributs de zoom (**zoom_x** et **zoom_y**) sur **1**.

Quelques ajustements — Point d'ancrage et décalage

Jusqu'à présent, nous utilisions le centre de l'image comme point d'ancrage de la rotation. Cependant, vous pouvez choisir n'importe quel point pour cela. Le centre de rotation est défini avec les attributs **anchor_x** et **anchor_y**. Présentement, ces attributs sont définis sur 160 et 120 (la moitié des dimensions de l'image), mais vous pouvez les modifier dans les boîtes de *nombre* intitulées ***Anchor point***.

• Essayez différents points d'ancrage et faites glisser la boîte de *nombre* ***d'angle de rotation*** pour voir l'effet. Certains paramètres de point d'ancrage que vous pouvez essayer sont *0,0* ou *40*, *30* ou *160*, *-120* ou *320, 240*. Vous souhaiterez peut-être définir l'attribut **boundmode** sur **1** afin de pouvoir voir plus clairement les effets de différentes rotations. Notez que les valeurs **anchor_x** et **anchor_y** sont spécifiées par rapport au coin supérieur gauche de la matrice, mais elles peuvent dépasser les limites des dimensions de la matrice.

En outre, vous pouvez déplacer l'image vers un emplacement différent dans la matrice de sortie après le zoom et la rotation, à l'aide des attributs **offset_x** et **offset_y**.

• Pour mieux voir cela, cliquez d'abord sur la boîte de *message* au-dessus de l'objet *pvar* dans le coin inférieur droit du patch. Cela ramènera l'angle de rotation, le mode limite, le zoom et les points d'ancrage aux paramètres que nous avons utilisés au début de ce chapitre. (Nous avons donné des noms aux objets d'interface utilisateur appropriés afin de pouvoir communiquer avec eux via *pvar*.) Définissez maintenant la boîte de *nombre* de zoom sur une valeur comprise entre 0 et 1 pour effectuer un zoom arrière sur l'image.

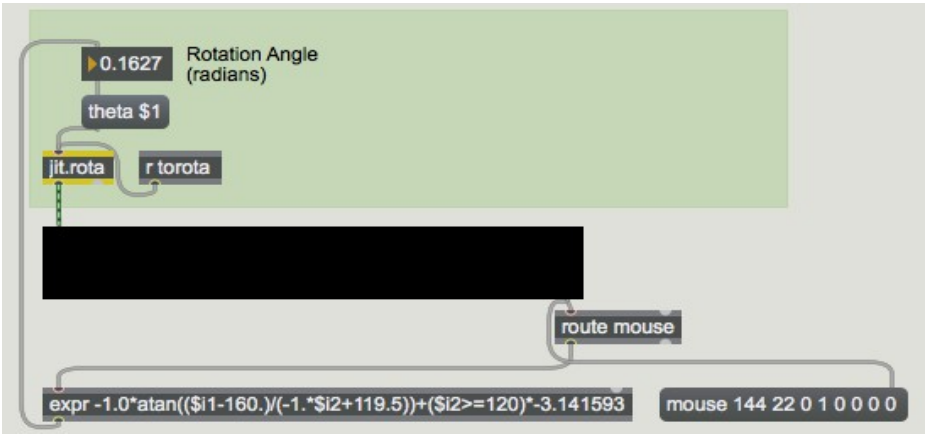
• Utilisez les boîtes de *nombre* de ***Location offset*** pour déplacer l'image en modifiant les valeurs **offset_x** et **offset_y**. Essayez ceci en conjonction avec **boundmode 4**, pour voir son utilité dans le mode "kaléidoscope".
• Lorsque vous avez terminé, réinitialisez les valeurs de ***Location offset*** sur 0.

Contrôle de rotation par la souris

Nous avons mis au point un autre moyen de faire pivoter l’image.

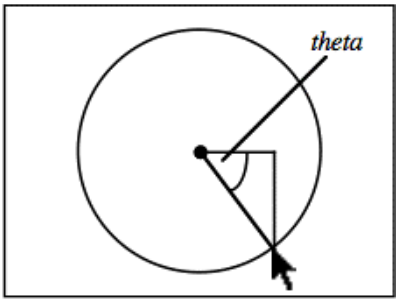
• Cliquez sur l'objet d'affichage *jit.pwindow* et, tout en maintenant le bouton de la souris enfoncé, faites glisser un petit mouvement circulaire autour du centre de l'objet.

jit.pwindow suit les mouvements de votre souris et, tant que le bouton de la souris est enfoncé, il envoie les informations de coordonnées (et autres informations de la souris) à la sortie droite sous la forme de messages **mouse**. Les deux premiers arguments du message **mouse** sont les coordonnées x et y de la souris, par rapport au coin supérieur gauche de *jit.pwindow*. Nous utilisons ces coordonnées pour calculer l'angle de la souris par rapport au centre de *jit.pwindow*, et nous envoyons cet angle à *jit.rota* en tant qu'argument de l'attribut **theta**.



Vous pouvez utiliser l'emplacement de la souris dans *jit.pwindow* comme informations de contrôle.

Détail technique: Voulez-vous vraiment savoir comment nous avons effectué ce calcul? Si oui, lisez la suite.
Si nous considérons le point central de *jit.pwindow* comme le point d'origine 0,0 et si nous considérons l'emplacement actuel de la souris par rapport à celui-ci comme un point situé le long d'un cercle autour de l'origine, nous pouvons décrire un triangle rectangle sur ces deux points. En prenant l'arc tangente des coordonnées de la souris y/x, nous obtenons l'angle de la souris par rapport au centre de *jit.pwindow*.



Nous prenons donc les coordonnées x et y entrantes, et la première chose à faire est de les convertir de manière à ce qu'elles soient relatives au centre du fichier *jit.pwindow*. Nous faisons cela en soustrayant 160 de la coordonnée de la dimension x (donc les valeurs x vont maintenant passer de -160 à 160) et en multipliant la coordonnée y par -1 (les valeurs augmenteront à mesure que nous monterons au lieu de descendre) puis en ajoutant 119,5. à cela. (Si nous ajoutions exactement 120, chaque fois que nous aurions une coordonnée de 120 de *jit.pwindow*, nous essaierions de diviser par 0 dans *expr*, ce qui est une opération mathématique non définie.) Une fois que nous avons converti les coordonnées x et y, nous prenons l'arc tangente de y/x pour obtenir l'angle en radians, puis multiplions cette valeur d'angle par -1 pour que la rotation de la souris dans le sens des aiguilles d'une montre provoque la rotation de l'image dans le sens des aiguilles d'une montre.

Cette méthode ne fonctionne que sur une plage de 180°, car la fonction arc tangente ne peut pas faire la différence entre l'emplacement de la souris et son point opposé sur le cercle. (Le calcul de y/x sera le même pour les deux points.) Ainsi, chaque fois que la coordonnée y de la souris passe dans la moitié inférieure de la fenêtre *jit.pwindow*, nous ajoutons un décalage de -π à l'angle thêta pour distinguer ces emplacements de leurs homologues sur le côté opposé. (C'est la dernière partie de l'expression.)

Notez que cette expression ne fonctionne que par rapport au point 160, 120 dans le *jit.pwindow*. Si nous voulions créer une expression qui fonctionne pour le point central de ***toute*** taille de *jit.pwindow*, il faudrait obtenir les dimensions de *jit.pwindow* avec un

message **getsize** et utiliser les valeurs **size** comme variables dans notre expression. Comme le disent les livres de mathématiques: "Nous laisserons cela comme un exercice pour le lecteur."

Sommaire

L'objet *jit.rota* fournit un moyen simple de faire pivoter une image avec son attribut **theta**, en spécifiant un angle de rotation, en radians. Il fournit également un moyen simple de zoom in ou de zoom out d'une image avec ses attributs **zoom_x** et **zoom_y**. Vous pouvez modifier le point central de la rotation avec les attributs **anchor_x** et **anchor_y** et déplacer l'image obtenue dans la matrice de sortie avec les attributs **offset_x** et **offset_y**. Vous pouvez modifier la manière dont *jit.rota* traite les cellules de la matrice situées en dehors de l'image résultante avec l'attribut **boundmode**. En combinant toutes ces fonctionnalités, vous pouvez obtenir des effets de duplication d'images et de kaléidoscope en plus du simple zoom et de la rotation.

Le zoom et la rotation impliquent des calculs internes assez intensifs effectués par *jit.rota*, de sorte que ces opérations sollicitent considérablement le processeur de l'ordinateur. Il existe d'autres attributs, qui ne sont pas abordés dans ce didacticiel, qui vous donnent accès à pratiquement tous les coefficients de la formule de rotation, ce qui vous offre encore plus de possibilités de déformation et de rotation de l'image. Ceux-ci sont montrés dans la référence *jit.rota*.

Pour gérer le contrôle de nombreux attributs à la fois, vous pouvez concevoir des processus Max automatisés pour générer des valeurs d'attribut et/ou des contrôles interactifs pour modifier les valeurs à l'aide de gestes.