32-Rendu des destinations

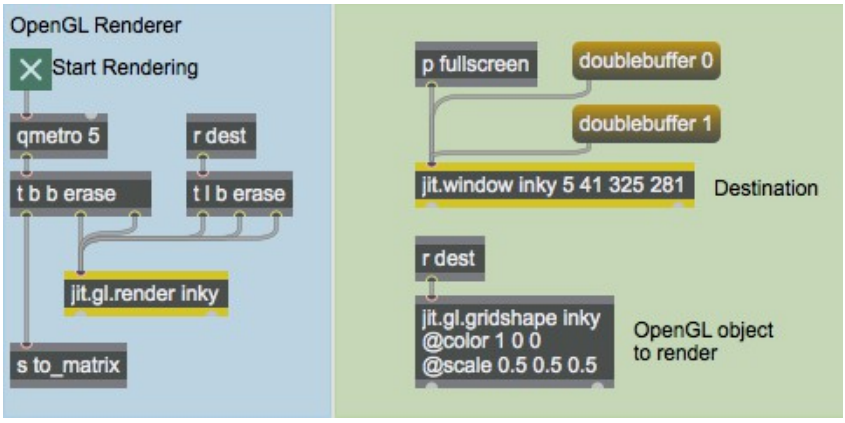
Dans le didacticiel précédent, nous avons vu comment dessiner des graphiques OpenGL dans un *jit.window* en utilisant l'objet *jit.gl.render*. Nous allons maintenant regarder les autres types de destinations dans lesquelles un objet *jit.gl.render* peut dessiner, à quoi les utiliser et comment basculer entre les destinations.

Ouvrez le didacticiel et double-cliquez sur le sub-patch **Render_Destinations** pour l'ouvrir.

Dans le coin supérieur gauche du patch se trouve un objet *jit.gl.render* avec l'argument **inky**, ainsi que d'autres objets.

• Cliquez sur le *toggle* dans le coin supérieur gauche du patch intitulé ***Start Rendering***.

Vous devriez voir une boule rouge apparaître dans l'objet *jit.window*. L'objet *jit.gl.gridshape* dessine la boule. Son premier argument, **inky**, spécifie le contexte de dessin actuellement dessiné par l'objet *jit.gl.render* dans la *jit.window*. Les autres arguments définissent les attributs de couleur et d'échelle de la boule.



*Dessinez une boule rouge dans la fenêtre nommée **inky**.*

Dessin et échange de tampons

Cliquer sur le *toggle* démarre l'objet *qmetro*, qui envoie des messages **bang** à l'objet *t* **b b erase**. Cet objet envoie d'abord un message **erase** suivi d'un message **bang** à l'objet *jit.gl.render*, puis un message **bang** qui est utilisé ailleurs dans le patch.

Lorsque l'objet *jit.gl.render* reçoit le message **bang**, il dessine tous les objets du groupe GL qui partagent sa destination, puis copie son tampon hors écran, le cas échéant, sur l'écran. ***Un tampon hors écran*** est une zone de mémoire non visible à l'écran, de la même taille que la destination du dessin. Par défaut, tous les contextes de dessin ont un tampon hors écran. Le dessin se fait dans le tampon, qui doit ensuite être copié sur l'écran pour être visible.

Le tampon hors écran permet de dessiner et d'animer sans scintillement, comme nous le voyons ici. Bien que le tampon soit effacé avant que la boule rouge ne soit dessinée à chaque fois, nous ne voyons jamais le tampon dans son état effacé. Pour voir à quoi ressemble le dessin sans la mémoire tampon hors écran, cliquez sur la boîte de *message* **doublebuffer 0** dans le coin supérieur droit du patch. Vous verrez probablement l'image commencer à scintiller. Cela se produit car le message **erase** provoque maintenant l'effacement de la fenêtre elle-même, et non du tampon hors écran. La fenêtre reste vide pendant une période courte mais indéterminée avant que la balle rouge ne soit dessinée, de sorte que vous pouvez voir la zone sous la boule vaciller entre le rouge et le gris foncé

de la fenêtre effacée. Cliquez sur la boîte de *message* **doublebuffer 1** pour renouveler le tampon hors écran et arrêter le scintillement.

Réglage du mode plein écran

Le sub-patch *p fullscreen* contient un objet *key*, un objet *select*, un *toggle*, une boîte de *message* et un objet *outlet* qui envoie les résultats du sub-patch à l'objet *jit.window*. Il s'agit d'une fonction standard de Max, donc nous ne l'examinerons pas trop en détail - le résultat est que la touche escape bascule entre l'envoi des messages **fullscreen 0** et **fullscreen 1** à l'objet *jit.window*. (Voir la section «Affichage plein écran» du *didacticiel 14*.)

• Appuyez sur la touche «esc» pour changer l'objet *jit.window* en mode plein écran et inversement.

> La touche escape semble être un moyen courant de basculer en mode plein écran, c'est pourquoi nous avons utilisé cette configuration dans de nombreux exemples de patchs. Il est important de noter, cependant, que ce n'est qu'une convention – il n'y a rien dans Jitter qui vous oblige à utiliser une touche ou une autre à cette fin.

Vous pouvez utiliser l'attribut **fsmenubar** en conjonction avec le message **fullscreen** pour masquer la barre de menus en plus de couvrir l'écran. Si l'objet *jit.window* a un attribut **fsmenubar** défini sur **0**, la barre de menus sera masquée.

Définir une destination jit.pwindow

L'objet *jit.gl.render* peut dessiner vers trois types de destinations différents. Le côté droit du patch contient un exemple d'objet de chaque destination: une *jit.window*, une *jit.pwindow* et une *jit.matrix*. Pour l'instant, nous effectuons le rendu vers l'objet *jit.window*. Pour changer la destination en l'objet *jit.pwindow*, nous devons d'abord nommer l'objet *jit.pwindow* puis définir les destinations de l'objet jit.gl.render et de l'objet *jit.gl.gridshape*.

• Cliquez sur le nom de la boîte de *message* **blinky** au-dessus des objets *jit.pwindow* de gauche.

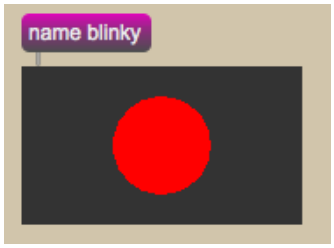
Cela nomme l'objet *jit.pwindow*, ce qui lui permet d'être utilisé comme destination de rendu. Pour basculer le dessin vers cette destination, nous devons envoyer des messages à la fois à l'objet *jit.gl.render* et à l'objet *jit.gl.gridshape*, leur indiquant leur nouvelle destination.

• Cliquez sur la boîte de *message* **blinky** dans la section ***Switch Destinations*** du patch.

Le symbole **drawto** est ajouté au message **blinky** et le résultat est envoyé à l'objet *s* **dest**. Deux objets reçoivent ce message - *jit.gl.gridshape* et *t* **l b erase**. L'objet *trigger* envoie une séquence de messages à l'objet *jit.gl.render*, qui lui indique de:

1- D'effacer la mémoire tampon de dessin de sa destination actuelle
2- Échangez ce tampon avec l'écran, effaçant visiblement l'ancienne destination, et
3- Basculez les dessins futurs vers la nouvelle destination.

Le résultat est que la boule rouge est affichée sur l'objet *jit.pwindow* à droite du patch.



Affichage de notre sortie rendue dans un objet jit.pwindow.

Définition d'une destination jit.matrix

En plus de dessiner sur les objets *jit.window* et *jit.pwindow*, nous pouvons dessiner sur les objets *jit.matrix*. Nous avons introduit les tampons hors écran dans la discussion sur les destinations *jit.window*, ci-dessus. Lorsqu'un objet 2 D *jit.matrix* est une destination de rendu, les données *jit.matrix* sont utilisées comme tampon hors écran. Cela place l'image d'une scène OpenGL dans le même format que les données vidéo, de sorte que tous les effets de traitement vidéo de Jitter peuvent être appliqués à l'image.

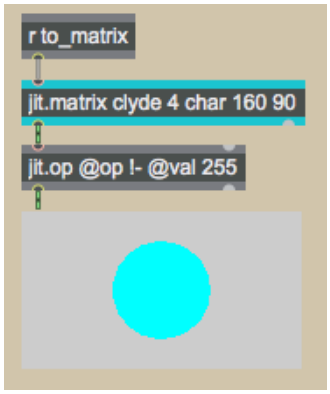
L'objet *jit.matrix* doit répondre à quelques critères pour que les graphiques OpenGL puissent y être dessinés:

- Il doit avoir quatre plans de données ***char***.
- Il doit avoir deux dimensions.
- Il doit être supérieur à huit pixels en largeur et en hauteur.

Nous avons une telle matrice dans le coin inférieur droit du patch du didacticiel. Elle mesure 160 pixels de large par 90 pixels de haut, soit les mêmes dimensions que l'objet *jit.pwindow* situé en dessous.

• Cliquez sur la boîte de *message* **clyde** dans la section **Switch Destinations** du patch pour dessiner dans l'objet *jit.matrix*.

Vous devriez voir une boule cyan sur un fond gris clair. Cela est dû au fait que l'image de la boule rouge générée par OpenGL est traitée via l'objet *jit.op*, qui soustrait chaque composant de couleur de 255, inversant ainsi l'image.



Rastérisation de la sortie dans une jit.matrix.

Il existe un autre détail important concernant le dessin sur les objets *jit.matrix*. Notez que sous l'objet *qmetro* se trouve un objet *trigger t* **b b erase**. Le message **bang** le plus à gauche (et donc le dernier) de cet objet est envoyé à l'objet *jit.matrix* dans lequel nous dessinons. Cela est nécessaire pour voir l'image dans la fenêtre *jit.pwindow*. Lorsque l'objet *jit.gl.render* reçoit un message **bang** ,

il termine la construction du dessin dans le tampon hors écran appartenant à l'objet *jit.matrix*. Mais pour envoyer la matrice résultante vers l'extérieur pour visualisation ou traitement ultérieur, il est nécessaire d'envoyer un message **bang** à l'objet *jit.matrix*.

> ***Rendu matériel vs logiciel***: l'un des grands avantages de l'utilisation d'OpenGL pour le rendu des graphiques est que la plupart du travail peut être effectué par l'accélérateur graphique de votre ordinateur, libérant ainsi le CPU pour d'autres tâches telles que la génération d'audio. Lorsque vous dessinez dans des objets *jit.window* ou *jit.pwindow*, le moteur de rendu matériel peut être utilisé. Malheureusement, en raison des limitations du logiciel système actuel, le moteur de rendu matériel ne peut pas dessiner directement dans les objets *jit.matrix*. Ce n'est pas une limitation inhérente à OpenGL, et cela pourrait changer à l'avenir. Cependant, pour l'instant, cela signifie que dessiner directement dans des matrices Jitter est significativement plus lent que dessiner sur des objets *jit.window* ou *jit.pwindow*, en particulier pour les scènes complexes ou celles impliquant des textures.

Rendus multiples et ordre de dessin

• Ouvrez le sub-patch **More_Render_Destinations**.

> Notez que les objets *jit.gl.gridshape* ont leur mise à échelle z fixée à 0. Cela en fait essentiellement des objets 2D.

Pour déplacer une scène OpenGL vers une destination différente, l'objet *jit.gl.render* ainsi que tous les objets du groupe GL impliqués dans le dessin de la scène doivent recevoir le message **drawto**. Pourquoi ne pas simplement envoyer un message au moteur de rendu, lui demandant de déplacer la scène entière? La raison est que chaque objet de groupe GL peut avoir une destination indépendante, ainsi que chaque moteur de rendu. Les objets du groupe GL peuvent être déplacés entre plusieurs moteurs de rendus. Pour voir un exemple de l'utilité de cette fonctionnalité, veuillez consulter la partie droite du patch de ce tutoriel.

À droite se trouvent trois objets *jit.pwindow* nommés A, B et C. Les objets de boîte de *message* au-dessus d'eux ne sont pas strictement nécessaires, car une fois qu'un objet *jit.pwindow* a été nommé, son nom est stocké avec lui dans le patch. Mais ils sont utiles ici comme étiquettes et comme rappel des messages à envoyer pour nommer les objets.

Il y a également trois objets *jit.gl.render* dans le patch. Chacun d'eux est dirigé vers une destination différente.

• Cliquez sur le *toggle* intitulé ***Start Rendering***.

Cela envoie de manière répétée le message **erase** suivi d'un message **bang** à chacun des trois moteurs de rendus. Vous devriez voir un cercle jaune dans un cercle bleu, dans la destination de dessin la plus élevée. Cette scène OpenGL simple est créée par les deux objets *jit.gl.gridshape* en bas à gauche du patch. Nous pouvons déplacer chacun de ces objets vers l'une des trois destinations de dessin présentes.

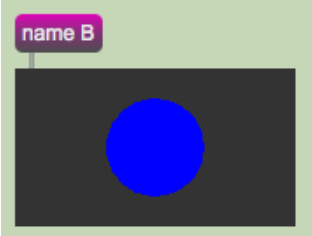
Rendu vers *jit.pwindow* A.

• Cliquez sur la boîte de *message* la plus haute **reading B** dans la section ***Switch destinations*** du patch du didacticiel. Cela change la destination du cercle bleu en contexte de dessin nommé "B" - il apparaît maintenant dans la fenêtre centrale de *jit.p*.

• Cliquez sur la boîte de *message* inférieure **reading B** dans la section ***Switch destinations*** du patch du didacticiel. Cela change la destination du cercle jaune en contexte de dessin nommé "B". Les deux objets sont réunis.

Chaque fois qu'un objet *jit.gl.render* reçoit un message **bang**, il dessine tous les objets du groupe GL qui ont été ajoutés à son contexte. Il dessine les objets dans l'ordre dans lequel ils ont été ajoutés au contexte. Dans ce cas, le cercle jaune a été ajouté au contexte de dessin nommé "B" après le cercle bleu, il apparaît donc en haut. Pour changer cet ordre, nous pouvons envoyer le message **drawto B** à l'objet *jit.gl.gridshape* qui dessine à nouveau le cercle bleu. Cela le retirera de sa place actuelle dans la liste des objets pour le contexte nommé B et le rajoutera à la fin de la liste.

• Cliquez à nouveau sur la boîte de *message* supérieure en lisant la section ***Switch destination*** du patch du didacticiel. Le cercle bleu devrait maintenant masquer le cercle jaune.



Le cercle bleu masque le jaune.

Sommaire

Nous avons introduit un système flexible pour créer plusieurs moteurs de rendus OpenGL et destinations de dessin, et pour déplacer des objets dans le groupe GL entre eux à l'aide de messages **drawto**.

Trois objets Jitter peuvent fonctionner comme destinations de dessin: *jit.window*, *jit.pwindow* et *jit.matrix*. Chaque type de destination a des utilisations différentes. Un objet *jit.window* peut être déplacé sur différents moniteurs et rapidement agrandi pour couvrir l'écran. Un objet *jit.pwindow* conserve son emplacement dans le patch. Un objet *jit.matrix* peut être utilisé comme tampon hors écran pour le rendu. La sortie de l'objet *jit.matrix* est une image tramée de la scène 3D, à laquelle un traitement vidéo supplémentaire peut être appliqué.