5: Ordre des messages et débogage

Introduction

Ce tutoriel est axé sur l'**ordre des messages** - la séquence dans laquelle les messages sont passés d'objet à objet et comment les objets les génèrent. Nous allons également utiliser certains des outils de **débogage** de Max pour déterminer le fonctionnement d'un patch.

Les patchs Max souvent donnent l'impression que tout se passe en même temps; en réalité, les messages sont produits et traités dans un ordre spécifique. Afin de créer des patchs qui fonctionnent correctement, nous devons comprendre l'ordre dans lequel les choses se produisent et comment contrôler les matrices complexes d'actions.

Pour ouvrir le patch du didacticiel, cliquez sur le bouton vert **Ouvrir didacticiel** dans le coin supérieur droit de la fenêtre de documentation.

De droite à gauche, de bas en haut

Jetez un coup d'œil au patcheur du tutoriel. Ce fichier contient un certain nombre de petits patchs que nous utiliserons pour apprendre les règles suivies par les messages. Cliquez sur le *bouton* le plus en haut dans le patch du haut à gauche; il semble que les trois objets bouton connectés se déclenchent simultanément. C'est une illusion: les messages sont envoyés par les câbles du patch dans un ordre séquentiel.

Le moyen le plus simple de voir cela en action consiste à utiliser certains des outils de débogage de Max. Sélectionnez **Activer le débogage** dans le menu **Déboguer** ou cliquez sur l'icône Activer le débogage (clé à molette) en bas de la fenêtre. Assurez-vous également que **Auto Step** est désactivé dans le menu **Débogage**. Maintenant, la rangée supérieure de patchs dans le tutoriel aura de petits cercles rouges avec des numéros couvrant les cordons de patch. Ceux-ci s'appellent des **points d'observation** ou simplement des **points d'arrêt**. Lorsque nous activons notre patcheur, l'opération **s'arrête** à chaque point d'arrêt. Nous pouvons examiner l'état des choses, puis continuer l'opération jusqu'au prochain point d'arrêt.

Cliquez sur le bouton le plus en haut dans le patch de gauche. Au lieu du clignotement immédiat de tous les objets *bouton* qui s'allument, le patch de connexion *le plus à droite* affiche un indicateur de chemin de message - un cercle vert animé. De plus, une fenêtre s'ouvre appelée **fenêtre de débogage**. La fenêtre de débogage nous indique qu'un message **bang** a été intercepté par un point d'arrêt; de plus, elle nous indique quel point d'arrêt a été déclenché (dans ce cas, le point d'arrêt n°1), le nom du patcheur et la classe de l'objet émetteur et récepteur (dans ce cas, les deux sont des objets bouton). Sélectionnez **Step** dans le menu **Debug** (ou cliquez sur le bouton Step dans la barre d'outils de la fenêtre Debug). Vous verrez que le cordon du milieu clignote; sélectionnez à nouveau Step et vous verrez que le cordon le plus à gauche clignote. Sélectionnez Step une fois de plus pour terminer notre trace de patch. Lorsque la sortie d'un objet est connectée à plus d'une entrée, les messages sont envoyés dans un ordre allant **de droite à gauche**.

Mais que se passe-t-il lorsque les objets récepteurs sont empilés verticalement (et donc à la même position horizontale) ? Le débogage étant toujours activé, cliquez sur le bouton le plus haut dans le patch central de la rangée supérieure. Lorsque les points d'arrêt se déclenchent, sélectionnez Step dans le menu Debug. Lorsque nous parcourons le patch, nous voyons que le bouton le plus bas reçoit son message en premier, suivi du milieu, puis du haut. Donc, nous pouvons voir que Max a deux niveaux d'ordre: de droite à gauche, puis de bas en haut.

L’ordre des messages a une autre particularité: c’est l’effet des messages qui entraînent la génération d'autres messages. Le troisième patch illustre ce problème. Activez **Autostep** pour celui-ci. Cliquez sur le bouton le plus en haut et observez comment les messages passent d'un objet à l'autre. Nous voyons que les messages se déplacent jusqu'au bout d'une chaîne de messages avant d'envoyer un message au prochain cordon de patch de la branche. Notez que la fenêtre de débogage vous montre toute la chaîne de messages d'une une branche; elle ne s'efface que lorsqu'une branche est terminée.

Pour résumer, les règles d'ordonnancement des messages dans Max sont les suivantes:

1. De droite à gauche ou de bas en haut pour les objets alignés verticalement.
2. Toutes les actions sur une branche sont terminées avant l'activation de la branche suivante.

Notez que, pour déterminer l'ordre de droite à gauche ou de bas en haut, c'est l'emplacement de l'entrée connectée, et non le chemin des câbles de raccordement, qui détermine l'itinéraire du message.

A titre d’exercice, créez une nouvelle matrice d’objets bouton, avec des tableaux verticaux et horizontaux d’objets bouton (en utilisant éventuellement l’option **Aligner** du menu **Arrangement**), ainsi que plusieurs profondeurs d’objets. Utilisez les boîtes de *commentaires* pour numéroter les boîtes dans l’ordre dans lequel vous pensez qu'elles seront déclenchées, puis tracez le patch. Faites-le jusqu'à ce que la commande de message devienne une seconde nature !

Sélectionnez **Désactiver le débogage** dans le menu **Débogage**, puis examinons les autres patchs du patcheur du didacticiel. (Pour plus d'informations sur l'utilisation du débogueur, voir la vignette de *débogage*.)

Plus d'ordonnancement des messages

Parfois, il n'est pas pratique de faire correspondre l'ordre spatial du patch au résultat souhaité. Dans la rangée inférieure des patch, le patch le plus à gauche est bien structuré: il est bien agencé et présente des *messages* de nombres numérotés par ordre croissant de gauche à droite. Si nous voulions que les messages passent du plus bas au plus haut, ce serait faux; lorsque nous cliquons sur le *bouton* du haut (bouton A), nous voyons les chiffres sortir dans l'ordre inverse dans la console Max. Cela est dû au fait que les boîtes de *messages* reçoivent des messages **bang** dans l’ordre de droite à gauche.

Le prochain patch montre une version corrigée de ce patch, utilisant un nouvel objet: *bangbang*. Cet objet prend un message entrant et produit des messages **bang** à partir de ses sorties, dans l’ordre de droite à gauche. Le nombre de sorties est déterminé par l'argument de bangbang. Les sorties sont connectées aux boîtes de messages avec des cordons de raccordement croisés; lorsque nous cliquons sur le bouton B, la Console Max affiche les messages dans l’ordre souhaité.

Remarquez que les sorties de *bangbang* se déclenchent dans un ordre de droite à gauche, imitant le l'ordre des messages des cordons de branchement. La plupart des objets dotés de plusieurs sorties respectent cette règle: les sorties sont produites dans l'ordre, de la sortie la plus à droite à la sortie la plus à gauche.

L'utilisation de l'objet *bangbang* rend l'ordre des messages explicite; en d’autres termes, il force les messages à suivre un chemin spécifique, quelle que soit leur orientation spatiale, et nous permet de placer des objets n’importe où dans le patch, tout en sachant qu’ils seront déclenchés dans un ordre bien défini en fonction des sorties à partir desquels nous les déclenchons. Un autre objet qui fournit

un ordre explicite est l'objet *trigger*. L'objet trigger accepte toute entrée et génère des messages en fonction de ses arguments. Les arguments déterminent le nombre de sorties et définissent leur type, avec les options **l** pour list, **b** pour bang, **i** pour entier, **f** pour nombre à virgule flottante et **s** pour symbole (un message texte). Vous pouvez également utiliser des entiers spécifiques, des nombres à virgule flottante ou des symboles comme sorties constantes.

Le patch suivant contient deux boîtes de *messages* avec l'entier **90**. Si nous cliquons sur celle étiquetée C, qui est directement connectée aux objets *imprimer*, la Console Max affiche l'ordre typique des messages de droite à gauche. Cependant, si nous cliquons sur la boîte de messages (D) connectée à l'objet *trigger*, nous voyons que nous pouvons acheminer les cordons de patch vers *l'impression* dans l'ordre de gauche à droite (inversé). Les messages envoyés sont tous imprimés sous la forme de l'entier 90, car nos arguments pour le trigger sont de type **i**.

Le prochain patch à droite utilise également le message entier **90** et est connecté à un objet *trigger*. (Notez que le nom "trigger" peut être abrégé "**t**"). Cependant, dans ce cas, trois arguments différents sont utilisés. Lorsque nous cliquons sur la boîte de message E, nous voyons que les sorties des trois sorties sont toutes différentes, ce qui correspond à la différence des arguments. La sortie la plus à droite, avec l'argument de type **i**, produit une sortie inchangée **90**. La sortie du milieu, dont l'argument est de type **f**, convertit le message entrant à son équivalent à virgule flottante 90.000000. Enfin, la sortie la plus à gauche, de type **b**, transforme le message entrant en un bang. Ainsi, en plus de rendre explicite l'ordre des messages, l'objet trigger peut effectuer une conversion de type des messages.

Le patch le plus à droite montre l'utilisation de valeurs constantes dans l'objet *trigger*. Lorsque nous remplaçons la boîte à virgule flottante F par un nombre quelconque, l'objet trigger convertit à nouveau le nombre entrant, mais envoie également un certain nombre de symboles utilisés comme valeurs constantes. Nous constatons également que le message entrant ne modifie pas la sortie de la sortie du **bang** - un bang est un bang, quel que soit le type de message entrant.

Ce serait le bon moment pour créer d'autres matrices de *boutons*, en les combinant avec des objets *trigger* et *bangbang*, et en les étiquetant avec l'ordre attendu des messages.

Le problème des boucles

Le patch final de notre fichier de didacticiel est un ensemble de quatre objets *boutons*, tous interconnectés dans une boucle. Lorsque nous cliquons sur l'un des objets boutons, le système s'arrête avec une erreur de **débordement de pile**. Pourquoi cela s'est-il produit ? Cette construction circulaire est une boucle de rétroaction. Un bouton envoie au suivant, qui envoie au suivant, qui envoie au suivant, qui envoie au premier, déclenchant le cycle à nouveau. Si on les laissait continuer, ces actions figeraient Max, nécessitant un **arrêt forcé** et la perte de tout travail non sauvegardé. Lorsque cette situation se produit, Max ferme sa programmation pour empêcher que quelque chose d'autre ne se produise; une fois que nous avons trouvé le problème et que nous l'avons corrigé dans notre patch (par exemple, en déconnectant l'un des objets bouton de la boucle), nous pouvons réactiver le patch en effaçant le message de débordement de pile situé en haut du patch.

Résumé

Une compréhension approfondie des règles d'ordonnancement des messages est nécessaire pour créer des patchs fonctionnant correctement. La règle de transmission de messages est la règle implicite de droite à gauche et de bas en haut, mais vous pouvez utiliser des objets tels que *bangbang* et *trigger* pour rendre la commande explicite. Dans tous les cas, cependant, vous devez

faire attention aux boucles de rétroaction - ce sont des tueurs de patchs au sens littéral !