16 : Messages à distance

introduction

Jusqu'à présent, dans ces tutoriels, tous les messages entre objets ont été envoyés à l'aide de cordons de raccordement. Ici, nous explorons une autre façon de transférer des messages entre objets - en utilisant le mécanisme de messagerie à distance intégré à Max. Il existe des objets spécifiques pour cela, appelés envoyer (*send*) et recevoir (*receive*), ainsi que des méthodes pour envoyer le contenu d'une boîte de *messages* à un objet. Enfin, nous utilisons certains objets qui permettent le stockage de données variables et peuvent partager des valeurs à travers des patchs complexes.

À mesure que nos conceptions deviennent plus complexes, nous devons tirer parti de davantage de techniques pour modulariser et compartimenter nos patches, ce qui les rend plus faciles à comprendre, à entretenir et à étendre. Lorsque nos patchs contiennent de nombreux objets, de longues connexions sinueuses de cordons de raccordement peuvent entraîner une complexité visuelle inutile. L'utilisation d'outils permettant le passage de messages sans cordons de raccordement peut nettoyer nos patchs sans en changer le fonctionnement.

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

Jouer avec *envoyer* et *recevoir*

> *Note technique : De nombreux objets du patch du didacticiel sont colorés pour vous aider à trouver des paires d'objets apparentés, mais la couleur n'a rien à voir avec le routage des messages.*

Le moyen le plus courant d'envoyer des messages sans cordon de raccordement consiste à utiliser les objets d'*envoi* et de *réception*. Si nous examinons le patcheur du didacticiel, la partie supérieure gauche du patch montre une paire d'objets d'*envoi* et de *réception* qui transmettent une valeur entière. Lorsque nous changeons la boîte de *nombre* supérieure, la valeur entière est transmise à l'objet d'*envoi* **fred**. Cette valeur est «diffusée» dans l'environnement Max et peut être «reçue» par n'importe quel objet de *réception* portant le nom **fred**. Dans ce cas, l'objet de *réception* **fred** situé immédiatement en dessous reçoit le message et l'envoie pour qu'il soit affiché dans la boîte de *nombres* jointe.

Puisqu'un message est diffusé par un ***nom***, tout objet de *réception* partageant le ***nom*** de l'objet d'*envoi* peut recevoir le message. Par conséquent, vous pouvez avoir plusieurs objets de *réception* pour un message et plus d'un objet d'*envoi* transmettant des messages en utilisant un seul ***nom***. C'est exactement ce que fait, dans notre patch de test, le système de curseur connecté à l'objet d'*envoi* **bob** : il envoie la valeur (définie à l'aide du *curseur* ou de la boîte de *nombre*) à tous les objets de *réception* des messages **bob**. En bas à droite du patch se trouvent deux objets **bob** de *réception* différents : l'un connecté à un système à *curseur*, l'autre est directement connecté à une boîte de *nombres*. La modification de la valeur du *curseur* supérieur modifie simultanément les deux valeurs de réception.

Lorsque vous travaillez avec des paires d'objets d'*envoi* et de *réception*, il est important de réaliser que le message sera saisi par n'importe quel objet de *réception* n'importe où dans l'environnement Max, même dans un autre patcheur. Pour tester cela, ouvrez un nouveau patcheur (en sélectionnant **New Patcher** dans le menu **Fichier**). Ajoutez un nouvel objet et entrez *receive* **bob**. Connectez la sortie de l'objet de *réception* à une boîte de *nombres*, puis revenez au patch du didacticiel. Si vous modifiez notre *curseur* de test, vous verrez la sortie reçue dans le nouveau patch ainsi que tous les autres objets de *réception* **bob** dans le patch principal.

Lorsque vous travaillez avec les patches d'autres personnes, vous pouvez voir des objets nommés «**s**» ou «**r**». Comme les objets d'*envoi* et de *réception* sont très populaires, ils portent respectivement les noms d'alias (ou raccourci) de «s» et de «r». En plus d'être plus faciles à taper, ces noms raccourcis vous permettent également de garder l'objet petit !

Travailler avec ; et forward

Une autre utilisation courante de la messagerie à distance consiste à envoyer le contenu d'une boîte de *messages* à un ou plusieurs objets de *réception*. Plutôt que de devoir toujours connecter les boîtes de *messages* aux objets d'*envoi*, il existe un format spécial de boîtes de *messages* qui permet d'envoyer directement des messages aux objets de *reception*. Cela implique de placer un point-virgule (**;**) et le nom d'un objet de *réception* devant le message. Ainsi, **; bob 155** équivaut à envoyer la valeur **155** à tout objet de réception nommé **bob**.

La grande boîte de ***messages*** dorée en bas à gauche du patch est un exemple d'envoi direct de message. Quatre messages sont générés: deux sont envoyés aux destinations **lcd1** et deux aux destinations **lcd2**. Si vous cliquez sur la boîte de *message*, vous verrez que les deux objets *lcd* reçoivent chacun les commandes de dessin appropriées par le biais de leurs objets de *réception* respectifs - le *lcd* supérieur (**lcd1**) dessine un cercle bleu ; le *lcd* inférieur (**lcd2**) dessine un rectangle vert.

Nous pourrions également avoir besoin d'envoyer des messages à l'une ou plusieurs des destinations en fonction de critères logiques. Cela peut être fait en utilisant la logique de commutation (par exemple, l'objet *gate* connecté à différents objets d'*envoi*). Cependant, nous pouvons également utiliser l'objet *forward* pour acheminer des messages vers des emplacements sélectionnés. Juste au-dessus de notre exemple de boîte de *messages* verte se trouve un objet *forward* avec une logique de soutien.

Vous pouvez sélectionner la destination nommée qui doit recevoir les messages de *forward* : le message ***envoyé***, envoyé à l'objet *forward*, définira le nom de la destination pour tous les messages entrants. Une fois qu'une destination a été définie, tous les messages seront acheminés de manière appropriée. Si nous cliquons sur le message **send lcd1**, les modifications apportées à la boîte de *nombre* entraîneront la création d'un rectangle coloré dans le premier objet *lcd*. Si nous cliquons sur le message **send lcd2**, la destination de sortie de l'objet *forward* est modifiée pour devenir l'objet de *réception* **lcd2**. Désormais, toute modification apportée à la boîte de *nombres* entraînera la création d'un rectangle dans le deuxième objet *lcd*.

Par un symbole similaire, un objet de *réception* ***sans*** argument peut être utilisé pour recevoir des messages de n'importe quel objet *envoi* nommé ou d'un *forward* ou de toute boîte de *messages* configurés de manière appropriée. Le message **set** adressé à l'objet de *réception* lui permet de "changer de nom", pour ainsi dire, afin d'écouter différentes parties du patcheur. À côté de notre objet *forward* se trouve un tel objet de *réception*. En le faisant basculer entre écoute de **bob** et de **fred** (à l'aide des boîtes de *messages*), puis en manipulant la boîte de *nombres* et l'objet *curseur* qui sont envoyés à ces noms, nous pouvons voir comment cela fonctionne. En attribuant à l'objet *réception* un **nom** qui n'existe pas (par exemple, ***aucun***) cela le déconnectera de toute écoute dans notre patch. Notez que pour qu'un objet de *réception* soit "commutable", il doit être créé ***sans*** argument (sinon, il n'a pas d'entrée).

Les objets *int*, *float* et *value*

Parfois, lorsque nous travaillons en programmation patc, nous avons besoin d'un emplacement pour stocker une valeur jusqu'à ce que nous ayons besoin de l'utiliser. Bien que nous puissions utiliser

des boîtes de *nombres* entiers et à virgule flottante, ces objets offrent la possibilité de demander à l'utilisateur de modifier la valeur via l'interface utilisateur. Afin de fournir un stockage et un rappel simple des variables, il existe un ensemble d'objets spécifiquement utilisés à cette fin. Les objets *int* et *float* peuvent servir de stockage temporaire de valeurs; les valeurs entrant par l'entrée ***gauche*** sont stockées et envoyées immédiatement par la sortie, tandis que les valeurs entrant par l'entrée ***droite*** sont stockées sans sortie. Si vous souhaitez récupérer la valeur stockée dans un objet *int* ou *float*, vous devez envoyer un message **bang** (par exemple, à partir d'un objet *button*) dans l'entrée gauche.

L'un des avantages des objets *int* et *float* est que vous pouvez initialiser la valeur sans utiliser d'autre message. L'argument fourni avec l'objet lui donne une valeur initiale ; il n'est pas nécessaire de les configurer avec des messages *loadbang* ou d'autres fonctions. Si nous regardons les objets *int* et *float* à droite de notre patch, nous pouvons voir comment ils fonctionnent. Ils n'envoient pas leur valeur lorsque le patch est chargé. Envoyer un **bang** avec les objets *button* produit leur valeur initialisée. Une fois que nous avons commencé à entrer des nombres dans les boîtes de *nombres*, ces nouvelles valeurs sont stockées à l'intérieur des objets jusqu'à ce que nous envoyons un **bang** de nouveau.

Dans notre patch, juste en dessous des objets *int* et *float*, nous trouvons une paire d'objets appelée *value*. L'objet *value* est comme une combinaison d'un objet *int* / *float* et d'un couple *envoi* / *réception* - chacun des objets *value* a un ***nom*** (dans ce cas, **joe**), et toutes les valeurs placées dans l'un des objets sont partagées avec tous les autres objets *value* portant ce ***nom***. Une chose à noter est que l'envoi d'un message dans un objet *value* ne déclenche pas la sortie - vous devez forcer explicitement la sortie en envoyant un message **bang** à l'objet. Dans notre patch de test, modifiez la valeur d'un objet puis envoyez un **bang** à l'autre: vous verrez que la valeur est transférée comme par magie de l'un à l'autre.

Contrairement aux objets *int* et *float*, l'objet ***value*** peut recevoir des messages de ***tout*** type ; des symboles, des entiers et des flottants peuvent tous être stockés par l'objet *value*, tout comme une liste. Bien que cela soit pratique, cela signifie également que vous devez faire attention aux objets que vous connectez à un objet *value* pour afficher la sortie. Notre patch de test a des boîtes de *nombres* connectées aux objets *value* ; toutefois, il serait peut être préférable de connecter un objet de *message* (par l'entrée droite) ou un autre objet d'affichage qui peut accepter un ensemble diversifié de valeurs.

Résumé

Pouvoir envoyer des messages sans avoir à connecter de cordons de patch peut nous donner plus de flexibilité dans la disposition des patchs, mais nous permet également de produire et de router des messages dans des patchs très complexes. Les objets *d'envoi* et de *réception* sont le moyen le plus courant de faire passer des messages dans un patch, mais l'utilisation de la notation **;** dans les boîtes de *messages* et l'utilisation de l'objet *forward* pour le routage peuvent contribuer à minimiser la complexité visuelle de notre travail. Enfin, nous avons vu comment les valeurs peuvent être stockées dans les objets *int* et *float*, et comment un objet *value* peut être utilisé pour stocker et transférer les données qui lui sont envoyées. Tous ces objets et techniques sont importants pour travailler avec des systèmes de patchs complexes.