18: Collection de données

introduction

Ce tutoriel couvre deux des objets les plus utiles du monde Max : *coll* et *route*. L'objet *coll* permet la collecte indexée de données; vous pouvez le considérer comme une petite base de données rapide pour les données de message. L'objet *route* est également lié au concept d'indexation : il envoie les données d'une sortie spécifique en fonction de l'index (première entrée) d'une liste.

Les données ***indexées*** sont très utiles pour le routage complexe d'informations. Souvent, les données sont déjà indexées pour vous : les contrôleurs MIDI indexent normalement la valeur avec le numéro du contrôleur, tandis que les informations de note sont souvent indexées par le canal MIDI. L'utilisation de l'indexation pour les données peut également être utile lors de l'envoi de messages dans un patch complexe : en créant un système d'indexation, vous pouvez envoyer des données sans distinction et faire en sorte que l'objet *route* trouve et décode les données si nécessaire.

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

Manipulation de base de *coll*

Ce tutoriel contient plusieurs patchs de taille petite à moyenne. Nous allons commencer avec le patch situé en haut à gauche.

Le premier patch (intitulé **1**) introduit l'objet *coll*. Trois boîtes de *messages* y sont reliées. Si vous cliquez sur les trois, vous allez charger l'objet *coll* avec trois ensembles de valeurs. Le moyen le plus simple de voir ce qui est stocké dans le *coll* est de double-cliquer dessus. Cela ouvrira une fenêtre d'édition qui vous montrera que le premier nombre a été stocké sous la forme d'un index et le second nombre a été stocké en tant que valeur. Pour récupérer une valeur de manière programmatique, vous envoyez une valeur d'index dans l'entrée : la sortie sera la valeur (ou les valeurs) stockée(s) à cet emplacement d'index. Si vous utilisez la boîte de *nombres* pour envoyer un **0** à l'objet *coll*, il produira **20** (la valeur à l'emplacement d'index **0**) ; **1** et **2** produiront **-50** et **33** respectivement. Il s'agit de l'utilisation la plus élémentaire et la plus courante de l'objet *coll* - comme mécanisme de stockage pour des ensembles de valeurs indexées.

Le patch intitulé **2** est légèrement différent en ce qu'il utilise des symboles, plutôt que des nombres, comme indices. Cliquez sur les trois boîtes de *message* de stockage, puis double-cliquez à nouveau sur *coll* pour voir les données stockées. Les données sont similaires à cellles du premier fichier *coll*, mais les index sont des symboles (mots) plutôt que des nombres. Maintenant, si vous cliquez sur les boîtes de *message* à gauche (***foo***, ***bar***, ***biz***), l'objet *coll* donnera la valeur correcte pour chacun des symboles.

Le troisième patch (intitulé **3**) contient déjà des données déjà stockées. Si vous sélectionnez un nombre compris entre **0** et **3**, l'objet *coll* produira les données correspondant à cet index. Dans ce cas, plutôt qu'un nombre unique, le *coll* stocke une chaîne de symboles (dans ce cas, le monologue d'ouverture de Richard III de Shakespeare) ; tous les symboles sont affichés sous forme de liste. L'ajout d'un message **préalable** à la liste nous permet d'afficher la chaîne de valeurs dans une boîte de *messages*.

Afin de stocker les données dans le patcher (comme nous l'avons fait avec cet exemple), vous devez définir un attribut dans l'inspecteur de l'objet *coll*. Déverrouillez le patch et ouvrez

l'inspecteur de l'objet *coll* dans le patch **3**. Vous remarquerez que l'attribut intitulé «Enregistrer les données avec le patch» est sélectionné. Lorsque cet attribut est activé, toutes les informations stockées dans le *coll* seront écrites dans le fichier du patcheur lors de son enregistrement et seront immédiatement accessibles lors de la prochaine ouverture du patch.

Utiliser *coll* pour dessiner et jouer

La section centrale de notre patch (section **4**) utilise un *coll* pour contrôler à la fois une fonction de dessin et une fonction de lecture MIDI. Dans ce cas, le *coll* est chargé avec des données prédéfinies ; cependant, au lieu d'avoir été stockées dans le patcheur, ces données sont chargées à partir d'un fichier externe contenant les données d'un relevé EEG. Le nom du fichier est “eeg.txt”, et ce nom de fichier est utilisé comme argument pour le *coll*, de sorte que les données soient lues dans le *coll* lors de sa première initialisation.

Au sommet du patch se trouve un simple petit système de compteur auto-incrémenté. Chaque fois qu'un nombre est généré, il s'incrémente d'une quantité déterminée par la boîte de *nombre* située en haut à droite. Par défaut, l'incrément est un, puisque c'est ce que contient l'objet +. La lecture de l'ensemble de données à cette vitesse peut toutefois prendre un certain temps, car notre fichier de données contient plusieurs milliers de lignes. Si vous modifiez la boîte de *nombres* en **15** (ce qui signifie que vous incrémentez les étapes de **15** à chaque **bang** généré par le *metro*), vous verrez le motif EEG beaucoup plus clairement. La sortie de ce générateur de nombre est utilisée pour récupérer les données indexées du *coll*. Et est envoyée à la fois au subpatcher **drawit** et à un simple lecteur MIDI (*makenote* / *noteout*). Le lecteur MIDI utilise simplement le nombre (mis à l'échelle dans la plage MIDI comprise entre 0 et 127) comme note MIDI, tandis que le subpatcher **drawit** utilise la valeur entrante comme décalage vertical du cercle dessiné. Défilant. Démarrez le *metro* (avec le *toggle*) et remarquez comment différentes valeurs d’incrémentation pour le comptage modifient le déroulement des données EEG. Si vous devez redémarrer le fichier au début, la boîte de *message* étiquetée **0** réinitialisera la boîte *int* à 0.

Stocker des données dans un *coll* pour une utilisation ultérieure est le meilleur moyen de traiter les grands ensembles de données. Cette capture EEG, qui représente plus de 18 000 points de données, rendrait votre patcheur très volumineux s’il était stocké dans le fichier du patcheur. Au lieu de cela, les données peuvent être capturées dans un *coll*, sauvegardées dans un fichier texte (à l'aide du message ***write***) et lues dans le *coll* au moment de l'exécution (avec le nom du fichier comme argument ou avec le message **read**).

Le segment de patch suivant, étiqueté **5**, montre quelques astuces supplémentaires que *coll* a dans sa manche. Ce *coll* a un nom - **cues** - qui gère 8 cues différentes gérées par le reste du patch (double-cliquez sur l'objet *coll* pour voir ce qui est stocké à l'intérieur). Vous pouvez utiliser la boîte de *nombre* connectée pour envoyer les messages, mais vous pouvez également utiliser les trois boîtes de *messages* pour passer en revue les mémoires. Le message de **départ** vous amène au début du contenu du *coll*, tandis que les messages **suivants** et **précédents** avancent et reviennent en arrière dans les mémoires. Cela vous permet de parcourir *coll* en séquence, même si les entrées ne sont pas numérotées de manière séquentielle, et de revenir automatiquement au début lorsque la fin sera atteinte. C’est la manière la plus simple d’implémenter un parcours séquentiel du contenu d’un *coll*. Sans vous soucier de l'objet *route* pour l'instant, parcourez la séquence en utilisant les messages de **départ**, **suivant** et **précédent** et observez comment différentes actions sont déclenchées par la sortie de l'objet *coll.*

Il y a aussi un petit patch juste à droite, où un autre *coll* est également nommé "cues". Si vous double-cliquez sur ce *coll*, vous verrez qu'il a le même contenu. En fait, comme il porte le même ***nom*** que le *coll* précédemment affiché, il partage le contenu du *coll*- tout comme l'objet *value*

partageait des données par le biais d'une convention de dénomination . L'utilisation de *coll* nommés qui partagent leurs informations vous permet d'avoir un accès facile à un ensemble de données à différents endroits de votre patch, ou même au sein de subpatchers.

Acheminement des messages avec *route*

Le petit patch en bas à droite montre la fonctionnalité de l'objet *route* : lorsque vous cliquez sur les boîtes de *messages*, les messages sont comparés sur la base de leur identifiant (le premier élément de leur liste); si l'identifiant correspond à l'un des arguments de l'objet *route*, le message est dépouillé de son identifiant et les données restantes (dans ce cas, un entier) sont envoyées à la sortie. L'objet *route* a une sortie pour chaque argument, ainsi qu'une sortie finale pour les éléments qui ne correspondent pas. Ce patch montre ce qui se passe si un message non reconnu est reçu : le message **zeppo** n'est pas énuméré dans l'objet *route* et ne peut donc pas être routé. Plutôt que d'ignorer le message, il envoie l'intégralité du message à la sortie droite - éventuellement pour être utilisé par un autre objet *route*, ou peut-être pour être envoyé (comme dans le cas présent) à la Console Max avec l'objet *print*.

Le segment de patch **5** utilise également l'objet *route* pour prendre des messages du *coll* et les analyser pour effectuer des actions sur différents objets du patch. Comme nous l'avons vu dans le *coll*, les données contiennent un symbole (**m1**, **m2**, **m3** et **pick**) et un élément de données (**0/1** pour **m1**-**m3** et **bang** pour le symbole **pick**). Lorsque ce message est reçu par l'objet *route*, il utilise le premier élément (le symbole) pour déterminer le routage de sortie et envoie le reste du message via cette sortie. Par conséquent, un message qui commence par **m1** sera acheminé par la première sortie, tandis qu'un message commençant par **pick** sera acheminé via la quatrième.

Si nous utilisons les messages *start* et *next* pour parcourir les *coll*, nous verrons qu'une séquence d'actions est effectuée sur la base du premier segment du message. Les objets *metro* sont démarrés et arrêtés, et l'objet *random* est déclenché deux fois. Vous pouvez voir que l'objet *route* enverra le contenu des message balisés là où il sera utile.

Résumé

L'objet *coll* est un moyen puissant de travailler avec des messages et des données indexés. l'objet fournit des méthodes pour obtenir des messages spécifiques ou pour parcourir séquentiellement le contenu. Vous pouvez partager le contenu entre des objets *coll* nommés, ce qui vous permet d'accéder à ces données dans l'ensemble de votre patch. *coll* peut également lire ses données à partir de fichiers externes. Nous avons également vu comment l'objet *route* vous permet de suivre et de déplacer des messages en fonction de leur contenu, ce qui vous permet d'acheminer les données dont vous avez besoin vers différents objets en fonction dun index.