14: Encapsulation

introduction

Ce tutoriel couvrira le concept d'**encapsulation** dans Max. L'encapsulation nous permet de placer des sections de notre patcheur dans leur propre subpatch à l'aide de l'objet *patcher*. Cela nous permet de masquer des parties de la logique du patcheur que nous n'avons plus besoin de voir, afin de rendre nos projets plus lisibles et plus concis. Nous examinerons la création d'un *patcher* contenant des objets, ainsi que l'utilisation de certains outils d'édition pour encapsuler les parties d'un patcher qui n'ont plus besoin d'être vues.

Il est important d'encapsuler correctement votre logique Max pour garder vos fichiers de patch propres, faciles à lire et à conserver. La logique encapsulée peut être facilement éditée et peut également être sauvegardée comme des fichiers séparés pouvant être réutilisés dans d'autres patchs.

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

Encapsulation de base

En haut à gauche du didacticiel, deux petits patchs qui font la même chose - ils ajoutent tous deux **5** au **nombre** entrant. Le patch de gauche le fait avec un objet +, tandis que le patch de droite utilise un objet *patcher*. L'objet *patcher* (qui peut être abrégé "**p**", tout comme le *trigger* peut être abrégé "**t**") est littéralement un autre patch intégré dans le patch actuel. Pour voir le contenu d'un objet *patcher*, vous pouvez double-cliquer dessus (lorsque le patch est verrouillé) pour ouvrir la fenêtre de l'objet *patcher* encapsulé.

Si nous double-cliquons sur l'objet *patcher* **add5**, une petite fenêtre de patcher en affiche le contenu (dans une fenêtre intitulée **add5** - le nom du subpatch). C'est un simple subpatch, avec seulement trois objets. L'objet du milieu est évident - c'est l'objet + qui effectue l'addition. Les objets situés au-dessus et au-dessous de l'objet + sont des objets d'entrée et de sortie, respectivement. Un objet d'*entrée* fait en sorte que l'objet *patcher* qui le contient ait une entrée, tandis qu'un objet de *sortie* fait en sorte que le *patcher* qui le contient ait une sortie. De cette manière, nous pouvons créer des objets *patcher* qui agissent comme des objets Max intégrés. Plusieurs objets d'*entrée* et de *sortie* créeront des entrées et des sorties correspondantes sur l'objet *patcher* englobant, agencées spatialement en fonction de leur position dans le subpatch (par exemple, l'objet d'*entrée* le plus à gauche dans le subpatch correspondra à l'entrée la plus à gauche sur le *patcher*).

Lorsque vous travaillez avec des objets d'*entrée* et de *sortie*, il est utile de leur ajouter une ***assistance***. De cette manière, le *patcher* englobant peut afficher des informations utiles sur le type de message attendu par l'entrée ou produit par la sortie. Si vous sélectionnez une *entrée* ou une *sortie*, et que vous ouvrez ensuite l'inspecteur de l'objet, vous verrez qu'un champ de ***commentaire*** est disponible. Tout texte que vous entrez dans ce champ apparaîtra comme une aide dans l'objet *patcher* lors de l'édition du patch.

Effectuer une encapsulation

La création de subpatchers doit parfois intervenir après la mise en place de la logique. Peut-être que votre patcher s'est développé au-delà de ce que vous pensiez initialement, ou que vous avez ajouté une logique qui n'était pas prévue, et que les choses commencent à devenir compliquées. Dans ces cas, il serait utile de pouvoir sélectionner un groupe d'objets et de les transformer rapidement en un seul *patcher* encapsulé. Cela peut être fait en utilisant l'élément de menu **Encapsuler**.

Notre exemple de patch a un algorithme de dessin intéressant, dans lequel vous déplacez un palet autour d'une zone rectangulaire (en utilisant un objet appelé *pictslider*), et de petits cercles sont dessinés aléatoirement près de la position équivalente dans l'écran *lcd* au bas du patch. Si vous devez effacer l'écran *lcd*, vous pouvez appuyer sur la barre d'espace (caractère ASCII **32**, capturé par l'objet *key* et déclenché par l'objet *select*). La logique est plutôt désordonnée et cela ne nous aide pas vraiment de la voir se propager sur l'écran. Nous allons sélectionner une majorité de la logique et l'encapsuler dans un subpatch.

Il y a deux boîtes de *commentaire* («encapsulate from here» et «to here») que nous utiliserons pour sélectionner notre logique encapsulée. ***Déverrouillez*** le patch, puis sélectionnez (en cliquant et en faisant glisser) tous les objets entre et à droite des boîtes de commentaires. 15 objets seront sélectionnés. Sélectionnez **Encapsuler** dans le menu **Edition** et vous verrez que toute cette logique est repliée en un seul *patcher* sans nom. Si vous verrouillez le patch et double-cliquez sur l'objet *patcher* (ou commande double clic sur un patcher non verrouillé), le subpatch s'ouvre et affiche à nouveau tous les objets, liés au monde extérieur à l'aide de quatre objets d'*entrée* et un objet de *sortie*. Si, pour une raison quelconque, nous devions inverser ce processus ultérieurement, la commande **Dé-encapsuler** du menu **Edition** permettra de démêler le subpatch pour le ramener dans notre patcher principal, reconnectant tout correctement comme à l'origine.

Il est souvent utile de nommer nos objets *patcher*; bien que cela n'ajoute aucune valeur particulière au *patcher*, cela vous aidera (ainsi qu'aux autres) à comprendre la logique encapsulée dans l'objet. Pour le nommer, il suffit de cliquer à l'intérieur de l'objet et d'ajouter un premier argument qui est le nom nom que vous souhaitez utiliser. Quelque chose comme **draw_logic** serait parfaitement utile pour cet exemple.

Création de notre propre subpatcher

Lorsque vous savez à l'avance ce que vous souhaitez travailler dans un subpatch, il est facile de créer un objet *patcher* et de travailler dans sa fenêtre d'édition. En bas à gauche se trouve une paire de boîtes de *nombres* reliées par - rien. Nous allons faire un subpatch ici et l'utiliser pour convertir une valeur en une autre.

Créez un nouvel objet vide entre les boîtes de *nombres* en y plaçant la souris et en tapant "n". Dans la boîte du nouvel objet, tapez **p mycalc** - cela créera un nouveau *patcher* nommé **mycalc**. Une nouvelle fenêtre apparaîtra pour éditer notre nouveau subpatch. Commencez par une *entrée* et une *sortie* (disponibles sous l'icône **Ajouter un objet** dans la barre d'outils supérieure). Notez que l'ajout de ceux-ci au patcher ajoute une entrée et une sortie à l'objet *patcher* dans notre fenêtre principale. Ensuite, connectez l'*entrée* directement à la *sortie* - créant ainsi un objet «thru» qui ne fait rien d'autre que de transmettre le message entrant à la sortie. Testons cette encapsulation dans la fenêtre principale de patcher.

Maintenant que notre objet *patcher* **mycalc** a une *entrée* et une *sortie*, nous pouvons le connecter aux boîtes de *nombres*. Connectez la boîte de *nombres* du haut à l'*entrée* du patch; connectez la *sortie* du patcher à la boîte de *nombres* du bas. Verrouillez le patch et modifiez la boîte de *nombres* du haut: nous devrions voir la boîte de *nombre* du bas changer pour lui correspondre. Cela ne devrait pas être une surprise, car notre objet *patcher* encapsule une ligne droite entre les deux boîtes!

Revenons à la fenêtre d'édition du *patcher* et ajoutons un peu de logique - si la fenêtre s'est fermée, double-cliquez sur l'objet *patcher* pour l'ouvrir à nouveau et le déverrouiller. Commencez par sélectionner et supprimer le cordon de raccordement qui relie l'*entrée* et la *sortie*. Ajoutons maintenant un peu de math: créez deux nouveaux objets, un objet + **50** et un objet / **7**. Maintenant,

connectez l'*entrée* à l'objet +, le + à l'objet / et la sortie de l'objet / à la *sortie*. Nous avons changé notre encapsulation de **mycalc** qui n'était qu'un subpatch sans signification en un *patch* («entrée + 50) / 7». L'avantage de faire cela dans une encapsulation est que nous n'avons rien à changer dans le patch de niveau supérieur: toutes les connexions restent telles quelles. Maintenant, dans notre patch principal, le changement de la boîte de nombres supérieure produira le nouveau résultat sans aucune autre modification.

Résumé

L'encapsulation de la logique de patching accomplit deux choses:

1 -elle nous permet de travailler à des niveaux supérieurs sans avoir à interagir avec (ou même voir) tous nos objets; et,

2- elle nous permet de modifier notre logique de bas niveau sans avoir à changer quoi que ce soit à la programmation de haut niveau.

Lorsque vous utilisez des objets *patcher*, vous pouvez soit les créer et les remplir manuellement, soit sélectionner la logique de patcher existante et utiliser l'option de menu **Encapsuler** pour créer automatiquement un subpatch. Dans les deux cas, vous pouvez explorer plus avant et modifier le code du *patcher* sans autre modification du patch de niveau supérieur.