15: Abstractions

introduction

Dans ce didacticiel, nous étendrons le concept d'encapsulation pour inclure l'abstraction, c'est-à-dire la possibilité d'avoir la logique du subpatcher dans un fichier séparé et réutilisable que vous pouvez ensuite utiliser à l'intérieur de n'importe quel patch. Une fois enregistrées à l'extérieur dans un fichier séparé, les abstractions peuvent être modifiées pour utiliser des arguments afin de rendre un patch Max générique utile à votre application spécifique. En utilisant des abstractions pour vos tâches de programmation les plus utilisées, vous serez en mesure de réutiliser cette logique dans les projets futurs sans aucune duplication de travail.

Les abstractions sont essentielles pour soutenir votre connaissance et votre expérience accumulées de Max. Le mécanisme d'abstraction peut faire en sorte que vos subpatchs ressemblent et agissent comme des objets Max intégrés, et peut également accepter des arguments pour affiner sa fonctionnalité.

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

Une vue d'ensemble

Dans notre patcheur de tutoriel, vous verrez qu'il existe trois patchs différents. Dans ce cas, chacun fait exactement la même chose, mais à des niveaux d'abstraction différents. La première section (nommée **1**) est le patch entièrement visible. Si vous activez le *metro* à l'aide du *toggle*, nous voyons que le patch suit votre curseur autour de l'écran et dessine une version mise à l'échelle des mouvements de votre souris sur l'affichage *lcd*. Nous utilisons l'objet *bucket* dans ce patch; un exemple simple de l'objet est montré à gauche à côté du *lcd*. Dans sa forme la plus simple, lorsqu'il reçoit une valeur, *bucket* affiche toujours la dernière valeur qu'il a reçue. Cela le rend utile en tant qu'objet de retard à événement unique, que nous utilisons dans le patch principal pour construire des messages **linesegment** qui relient la position actuelle de la souris à sa position précédente.

Dans le cas de la section **1**, nous avons beaucoup de logique à l'écran et il peut être utile de l'encapsuler dans un subpatcher (comme nous l'avons fait dans le tutoriel précédent). Cependant, la logique du patcher qui interroge la souris et la met à l'échelle en fonction de la taille de notre écran semble être quelque chose qui pourrait être utile ailleurs; il serait intéressant d'en disposer pour d'autres patchs sans avoir à copier-coller d'un patch à l'autre.

C'est ici qu'interviennent les abstractions. Une abstraction est un *subpatcher* qui est sauvegardé en comme un fichier externe et peut être utilisé comme un objet Max standard. Tant que votre abstraction peut être trouvée dans le chemin du fichier Max, vous pouvez taper son nom dans une nouvelle boîte d'objet et elle sera chargée directement dans votre patch. Les abstractions utilisées dans ce tutoriel se trouvent dans le même dossier que le patcher qui y accède; une bibliothèque d'abstractions peut facilement être créée en les plaçant dans un dossier (ou dans un groupe de dossiers) à l'intérieur du dossier "patches" de votre installation Max, ou n'importe où qilleurs où Max cherche des fichiers.

Utiliser une simple abstraction

La deuxième section de notre patch (étiquetée **2**) montre une abstraction au travail. Désactivez le *metro* pour la section **1**, appuyez sur la barre d'espace (ce qui effacera l'écran *lcd*) et activez le *metro* pour la section **2**. Vous verrez que le programme fonctionne exactement comme avant.

L'objet appelé **WTHITM** (pour "Où est la souris?") est en réalité une ***abstraction*** de la logique de mise à l'échelle de la section **1**. Si vous double-cliquez sur l'abstraction **WTHITM**, une nouvelle fenêtre de patcheur affichera le contenu de cette abstraction. Notez que cela ressemble beaucoup à une encapsulation à une exception près - vous ne pouvez pas déverrouiller la fenêtre pour modifier le contenu. Ceci est dû au fait que **WTHITM** existe en tant que fichier patcheur séparé résidant dans le même dossier que notre patch de tutoriel.

Les abstractions sont destinées à être partagées entre plusieurs patchs. Par conséquent, vous ne voudriez pas modifier le contenu dans un patch, car cela pourrait compromettre ses fonctionnalités dans les autres patchs. Si vous souhaitez modifier une abstraction, vous devez ouvrir le fichier d'abstraction lui-même.

Bien qu'il existe plusieurs façons d'ouvrir le fichier d'abstraction, nous n'en mentionnerons que deux: nous pouvons ouvrir un **nouveau navigateur de fichiers** (à partir du menu **Fichier**) et taper **WTHITM** dans la fenêtre de recherche, puis double-cliquer sur le fichier patcheur qui apparaît. en réponse à notre requête, nous pouvons également cliquer sur l'icône **Modifier en lecture seule** (crayon) dans la barre d'outils d'abstraction. Quel que soit votre choix, ouvrez le fichier source **WTHITM**. Vous verrez à nouveau le contenu de l'abstraction, mais vous pouvez maintenant modifier la logique. Si vous modifiez quelque chose et enregistrez le fichier, tous les patcheurs qui utilisent cette abstraction refléteront les modifications, même si le patcheur est actuellement chargé. Vous pouvez voir cela en action en ajoutant une deuxième entrée à l'abstraction **WTHITM** et en la sauvegardant. Dès que Max voit une nouvelle version de l'abstraction, il la recharge et ajoute une nouvelle entrée à l'abstraction **WTHITM** dans votre patch de didacticiel.

Pendant que le patch **WTHITM** est ouvert, examinons la documentation qui a été ajoutée aux objets d'*entrée* et de *sortie*. Si vous ouvrez l'inspecteur de l'*entrée*, vous constaterez qu'il existe plusieurs types différents de documentation intégrée. Le premier s'appelle ***Annotation*** ; tout texte placé dans le champ d'annotation apparaîtra dans la fenêtre ***Indice*** **(Clue)** lorsque vous passez la souris sur l'objet. Le texte placé dans le deuxième champ de documentation, le champ **Trace (Hint)**, s'affiche dans une bulle d'indice lorsque vous survolez l'objet d'*entrée* dans un patcheur ***verrouillé***. Enfin, le texte placé dans la section ***Commentaire*** ne s'affiche pas dans le patcheur ; il est plutôt affiché comme texte d***'assistance*** lorsque vous survolez l'*entrée* de "l'objet" dans le patcheur de niveau supérieur. Il est sage de documenter vos objets d'*entrée* et de *sortie* dans des abstractions souvent utilisées, car cela peut fournir une documentation simple mais efficace pour votre logique réutilisable.

Utilisation d'une abstraction avec des arguments remplaçables

Dans notre abstraction **WTHITM**, aucune logique n'a besoin d'être renseignée sur le patcheur de niveau supérieur. Cependant, si nous voulons que l'abstraction mette correctement à l'échelle la sortie en fonction de la taille de notre objet *lcd* ? Dans ce cas, nous devons informer l'abstraction de la taille de l'écran *lcd*, ce que nous pouvons faire en utilisant des ***arguments*** à l'abstraction.

La section **3** du patch du didacticiel est similaire à la section **2**, mais l'abstraction utilisée s'appelle **WTHITM_scaled** et comprend deux arguments qui représentent les tailles horizontales et verticales de l'objet *lcd*. Si nous ouvrons et déverrouillons le patch "**WTHITM_scaled.maxpat**", nous verrons quelque chose de très intéressant: les facteurs de multiplication utilisent **# 1** et **# 2** comme espaces réservés pour les valeurs à fournir comme arguments. Comme vous vous en doutez, **# 1** est remplacé par la valeur du premier argument, tandis que **# 2** est remplacé par la valeur du second.

L'utilisation de ces valeurs remplaçables (appelées ***arguments dièse***) est essentielle pour créer des abstractions flexibles et réutilisables. Si vous aviez «319.» et «239.» codés en dur dans l'abstraction, vous ne pourriez utiliser que des objets *lcd* 320 x 240 pour afficher les mouvements de la souris. D'autre part, si vous forcez le patcheur de niveau supérieur à effectuer la mise à l'échelle finale (comme nous l'avons fait dans la section **2**), vous forcez le patch de niveau supérieur à dupliquer le travail pouvant être facilement résumé dans le patch de niveau inférieur. En utilisant des valeurs remplaçables dans vos abstractions, vous rendez le patcher de haut niveau est aussi simple que possible.

Résumé

En enregistrant votre logique dans une abstraction, vous pouvez créer des modules qui peuvent être utilisés dans le futur avec peu ou pas de programmation supplémentaire. Cela vous permet de tirer parti de vos connaissances de Max pour un travail plus efficace à l'avenir, et vous aidera à créer des systèmes de programmation qui sont modulaires et plus faciles à entretenir.