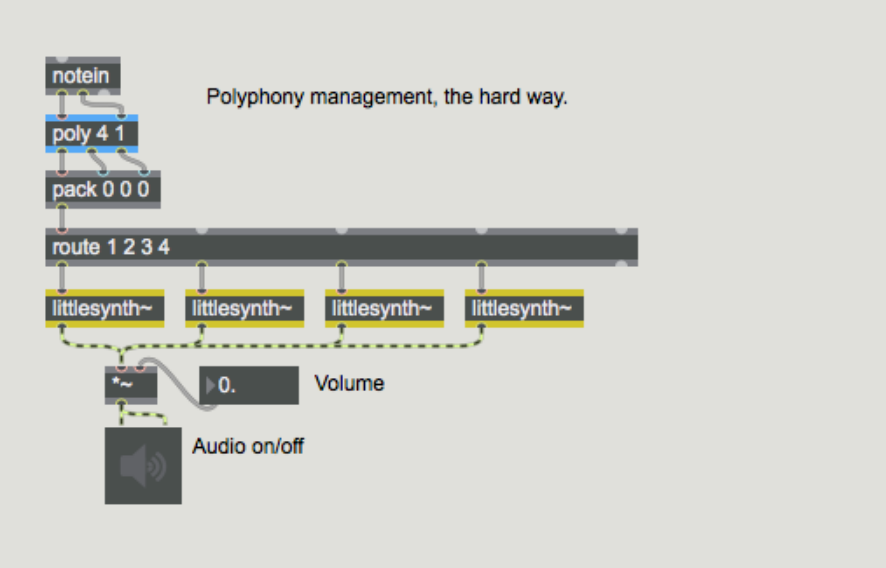
33-l'objet *poly~*

Une approche différente de la polyphonie

Dans un précédent tutoriel sur l'utilisation de MIDI avec MSP, nous avons montré comment utiliser l'objet *poly* pour effectuer des assignations de voix polyphoniques dans un cas simple. Ce chapitre va décrire une manière plus élégante et efficace de gérer l'allocation de voix polyphonique - l'objet *poly ~*.

Dans l'exemple du chapitre précédent, nous avons créé plusieurs copies de notre sub-patch de synthétiseur et utilisé la numérotation des voix de l'objet *poly* pour acheminer les messages vers différentes copies du sub-patch. Notre exemple aurait pu tout aussi bien utiliser n'importe quel type de sub-patch produisant du son.

• Jetez un coup d'œil au patcher du tutoriel. Ouvrez l'objet *patcher* intitulé 'thehardway'. L'exemple qu'il contient utilise le sub-patch **littlesynth ~** pour implémenter un simple synthétiseur polyphonique à quatre voix:



Bien que cette méthode fonctionne, elle présente deux inconvénients. Premièrement, il faut beaucoup de gestion interne pour dupliquer et patcher les multiples copies de **littlesynth ~** ensemble. Mais il y a aussi un problème en terme d'utilisation du CPU. Les quatre copies du sub-patch **littlesynth ~** sont toujours actives, traitant leur audio même même lorsqu'il n'y a pas de son produit.

Il existe un moyen de résoudre ce problème: l'objet *poly ~* vous permet de créer et de gérer plusieurs copies du même sub-patch MSP, au sein du même objet. Vous pouvez également contrôler l'activité de traitement du signal dans chaque copie du sub-patch afin de préserver les ressources du processeur.

L'objet *poly ~*

• Fermez le sub-patch "thehardway" et ouvrez l'objet *patcher* nommé "simple_poly". Activez l'audio dans le sub-patch en cliquant sur le message **startwindow** du *dac ~*. Cliquez sur l'objet *toggle* situé en haut du patcheur et augmentez le volume sur le curseur de *gain ~*.

L'objet *poly* ~ prend comme argument le nom d'un fichier patcheur, suivi d'un numéro qui spécifie le nombre de copies (ou instances) du patch à créer. Vous voudrez spécifier le même nombre de copies que celui que vous auriez dû dupliquer manuellement en l'implémentant la polyphonie à l'ancienne.

• Double-cliquez sur l'objet *poly* ~. Cela ouvre le sub-patch pour vous montrer l'intérieur de l'objet **littlebeep ~**.

Examinons le patch **littlebeep ~** pendant une minute. Bien que vous n'ayez jamais vu les objets *in*, *out~* ou *thispoly~* avant, le reste du patch est assez simple; il prend un numéro de note MIDI entrant, le convertit en une valeur de fréquence en utilisant l'objet *mtof* et sort une onde sinusoïdale à cette fréquence avec une durée de **140** millisecondes et une enveloppe d'amplitude fournie par l'objet *line* ~ pendant **140** ms avec une enveloppe activée.

Mais qu'en est-il des objets *in et out* ~ ? Les sub-patchs créés pour être utilisés dans l'objet *poly* ~ utilisent des objets spéciaux pour les entrées et les sorties. Les objets *in* et *out* créent des entrées et des sorties de contrôle, et les objets in ~ et *out* ~ créent des entrées et des sorties de signal. Vous spécifiez quelle entrée est affectée à quel objet en ajoutant un argument numérique à l'objet - l'objet *in* **1** correspond à l'entrée la plus à gauche de l'objet *poly* ~, et ainsi de suite. L'objet *poly* ~ garde la trace du nombre d'entrées et de sorties qu'il doit créer lorsque vous lui indiquez le sub-patch à charger.

Les messages envoyés à un objet *poly* ~ sont dirigés vers différentes instances du sub-patch de manière dynamique à l'aide des messages **note** et **midinote**, et manuellement à l'aide du message **target**.

Lorsque *poly* ~ reçoit un message de note dans son entrée de gauche, il parcourt les copies du sub-patch qu'il a en mémoire jusqu'à ce qu'il en trouve une qui n'est pas occupée, et lui transmet le message. Une instance de sub-patch peut dire à son objet *poly* ~ parent qu'elle est occupée en utilisant l'objet *thispoly* ~. L'objet *thispoly* ~ accepte soit un signal, soit un nombre dans son entrée pour définir son état d'occupation. Un signal nul ou une valeur de **0** envoyé à son entrée indique au *poly* ~ parent que cette instance est disponible pour les messages de note ou de midinote. Un signal ou une valeur non nulle envoyé à son entrée indique au *poly* ~ parent que l'instance est occupée; aucun message **note** ou **midinote** ne sera envoyé à l'objet jusqu'à ce qu'il ne soit plus occupé. L'état occupé a été prévu pour correspondre à la durée d'une note jouée par l'instance du sub-patch, mais il peut être utilisé pour signifier n'importe quoi. Dans l'exemple ci-dessus, lorsque le niveau audio sortant du * ~ est non nul - cette itération du sub-patch est actuellement occupée. Une fois que l'enveloppe d'amplitude de *line* ~ atteint **0** et que le son s'arrête, la copie de ce sub-patch thispoly ~ indique à *poly* ~ qu'il est prêt pour d'autres d'entrées.

Voix mutées

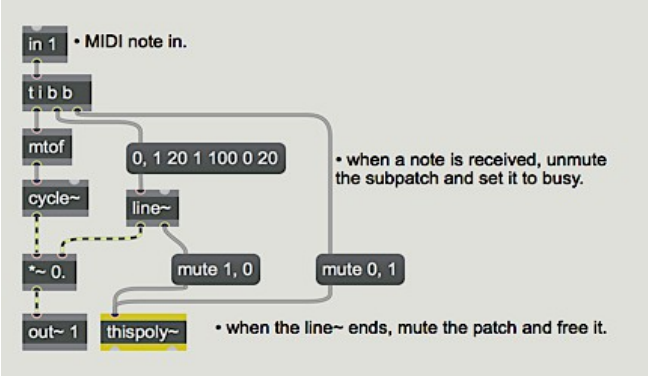
• Fermez le sub-patch 'simple_poly' et ouvrez l'objet *patcher* nommé 'poly_using_mute'. Démarrez-le comme vous l'avez fait pour le dernier patcheur et double-cliquez sur l'objet *poly* ~.

L'objet *thispoly* ~ peut également contrôler l'activité de traitement du signal dans chaque copie du sub-patch. Lorsque le message mute est envoyé à *thispoly* ~ suivi d'un **1**, tout traitement du signal dans ce sub-patch s'arrête. Lorsqu'un message **mute 0** est reçu, le traitement du signal recommence.

Dans ce patcheur, nous avons réécrit le sub-patch **littlebeep ~** pour en tirer parti en désactivant le traitement du signal lorsqu'une note est terminée et en la réactivant lorsqu'un nouvel événement est reçu. Bien que cela ne change pas la fonction du patch, ce serait plus efficace, puisque la quantité de

CPU allouée est toujours basée sur le nombre de notes en cours.

Pourquoi nous soucions-nous de l'efficacité? Eh bien, même si les ordinateurs modernes ont des performances inimaginables il y a quinze ans, ils ont toujours des limites. Le traitement audio repousse constamment ces limites, comme peu de choses le font. Lorsque la limite est atteinte, elle est audible sous forme de clics et de distorsion. C'est pourquoi il existe une option de sortie non temps réel. (Voir *E / S audio - Entrée et sortie audio avec MSP)* Maintenant, considérons ce patch:



Un patcher MSP silencieux

Comme indiqué, le patch ne fait aucun bruit. Dès qu'il recevra un numéro de note, il le fera, mais pour l'instant, il est en attente. Cependant, cela nécessite toujours un calcul. Le signal de l'objet *cycle* ~ est multiplié par **0**, donc rien n'est entendu, mais des multiplications se produisent 44,100 fois par seconde, sans parler de ce que *cycle* ~ fait pour produire un signal. Une fois que suffisamment de patches, plug-ins et autres sont ouverts, cela peut être «la goutte d'eau qui fait déborder le vase»; donc désactiver l'audio qui n'est pas actuellemnt nécessaire est une bonne idée. Le sub-patch **littlebeep2 ~** démontre l'utilisation du message mute à *thispoly* ~.Nous pouvons désactiver des voix individuelles dans *poly* ~ avec un message **mute** à l'objet *poly* ~ parent. Cela prend la forme de **mute n x** où **n** est le numéro de voix et **x** est **1** pour muté et **0** pour non muté.

Si nous envoyons à *poly* ~ le message **mute 0 1** tout traitement audio dans le *poly* ~ s'arrête. C'est un outil puissant pour la gestion les ressources DSP, même si vous n'avez besoin que d'une seule copie d'un processus. Il existe certains objets, comme *pfft* ~, (décrits dans *Signal Processing with pfft~*) qui peuvent être assez coûteux en calcul. Pourtant, les utilisations de *pfft* ~ sont suffisamment variées (changement de hauteur, filtrage précis et vocodage, entre autres) pour qu'un patch puisse facilement en comporter une douzaine. Cela suffit pour occuper 30% du CPU d'une machine de puissance moyenne. Encadrer chaque routine *pfft* ~ dans son propre *poly* ~ vous permettra de désactiver toutes celles qui ne sont pas utilisées actuellement. (Notez qu'une fois que vous avez désactivé tout le *poly* ~ avec **mute 0 1**, vous ne pouvez pas démarrer une seule voix avec quelque chose comme **mute 0 1** , avant d'avoir envoyé un **mute 0 0**.)

Cibler des voix individuelles

• Fermez le sub-patch 'poly_using_mute' et ouvrez celui nommé 'poly_using_target'. Lancez le programme et examinez la logique du patcheur à l'intérieur et à l'extérieur de l'objet *poly* ~.

Une autre façon d'allouer des événements en utilisant *poly* ~ est le message **target**. L'envoi d'un message **target** suivi d'un nombre entier dans l'entrée gauche d'un sub-patch *poly* ~ indique à *poly* ~

d'envoyer tous les messages suivants à cette instance du sub-patch. Vous pouvez ensuite utiliser *poly ~* en conjonction avec l'objet *poly* pour créer un synthétiseur MIDI.

Dans cet exemple de patch, des paires de hauteurs et de vélocités MIDI entrantes sont utilisées pour synthétiser un son sinusoïdal. Lorsqu'une liste est reçue, le sub-patch envoie un **bang** à *thispoly ~*, lui faisant sortir le numéro de l'instance ou de voix. Dans notre patcheur de tutoriel, le numéro de voix est envoyé par une sortie afin que vous puissiez le regarder depuis le patch parent.

Dans le patch parent, l'objet *poly* attribue des numéros de voix aux paires de hauteur / vélocité MIDI produites par *makenote*. Le numéro de voix de l'objet *poly* est envoyé à *poly ~* avec le message **target** en préambule, indiquant à *poly ~* d'envoyer les données suivantes à l'instance du sub-patcher **targetbeep ~** spécifié par *poly ~*. Lorsqu'une nouvelle note est générée, la cible change. Puisque *poly* garde la trace des suppressions de notes, il devrait recycler les voix correctement. La seconde sortie de *poly ~ rapporte* la voix qui a reçu le dernier message - elle devrait être la même que le numéro de voix émis par *poly*, car nous utilisons *poly* pour spécifier une cible spécifique.

Utilisation de *poly ~* pour le traitement audio.

• Fermez le patcheur 'poly_using_target' et ouvrez celui nommé 'poly_using_signal_input'. Activez le son, augmentez le curseur de *gain ~* et démarrez le *metro* en cliquant sur l'objet *toggle*. Changez la valeur dans l'entrée la plus à droite de **100.** à **50.** et écoutez le résultat.

La boîte de *nombres* à virgule flottante peut être utilisée pour spécifier des paramètres pour des instances spécifiques d'un sub-patch *poly ~*. En connectant un objet *loadbang* à *thispoly ~*, nous pouvons utiliser le numéro de voix pour contrôler la fréquence centrale d'un filtre.

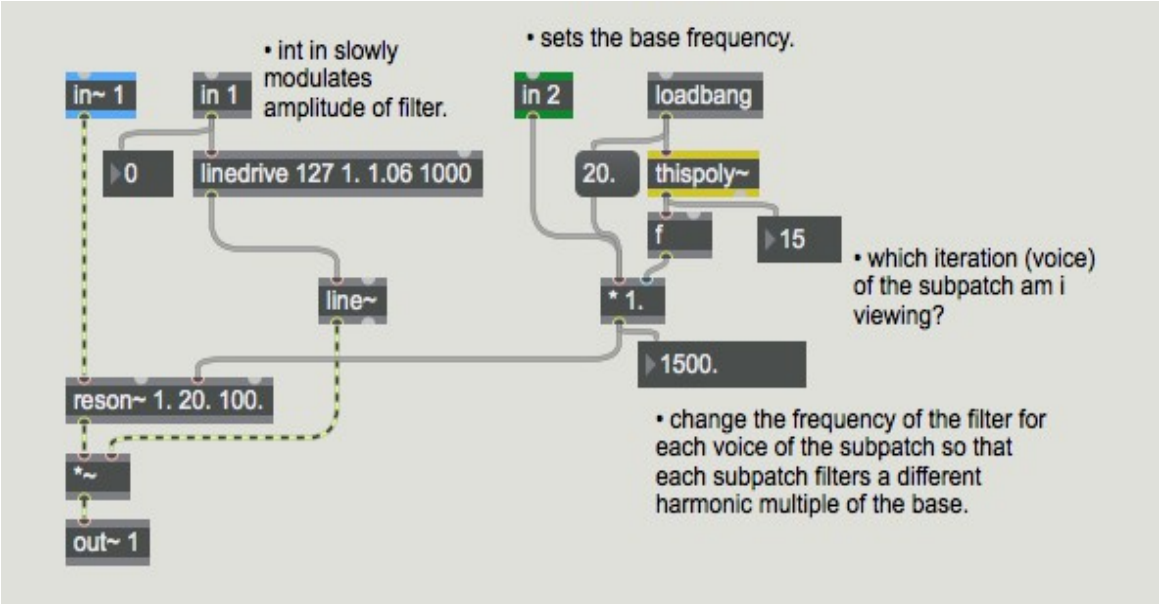
• Ouvrez le sub-patch **littlefilter ~** en double-cliquant sur l'objet *poly ~*.

L'abstraction **littlefilter ~** utilise le numéro de voix de *thispoly ~* et le multiplie par la fréquence de base reçue dans la deuxième entrée. Le signal entrant est filtré simultanément par les seize instances, l'amplitude de sortie de chaque instance étant contrôlée par un nombre entier entrant dans la première entrée.

L'objet *metro* du patcheur principal est relié à la fois à un *counter* et à un *random*. Le *counter*, qui alimente le message **target**, parcourt les 16 voix de **littlefilter ~** chargé dans l'objet *poly ~*, fournissant à chacune un nombre aléatoire permettant de contrôler l'amplitude de cette voix.

Un signal connecté à une entrée de *poly ~* sera envoyé aux objets *in ~* correspondants de toutes les instances de sub-patch, ainsi l'objet *noise ~* dans l'exemple ci-dessus alimente en bruit tous les sub-patchs à l'intérieur du *poly ~*. La deuxième entrée (qui correspond à la boîte *in* **2** du sub-patch) contrôle la fréquence de base des filtres. Notez que pour que la fréquence soit envoyée à toutes les itérations du *poly ~*, elle est précédée d'un message **target 0**. Vous pouvez ouvrir une instance spécifique d'un sub-patch *poly ~* en donnant à l'objet le message **open** suivi de la voix que vous voulez regarder.

• Ouvrez la voix **15** de l'abstraction de **littlefilter ~** en tapant ce numéro dans la boîte de *nombre* jointe au message **open**. Le patcher assigné à la voix numéro **15** devrait ressembler à ceci:



Comme vous pouvez le constater, la fréquence de base de cette itération particulière de **littlefilter ~** est 1500. Hz, qui est le multiple du numéro de voix (15) avec la fréquence de base la plus récemment entrée dans la deuxième entrée (100. Hz).

Résumé

poly ~ est un moyen puissant pour gérer plusieurs copies du même sub-patch pour l'allocation de voix polyphonique. L'objet *thispoly ~* fonctionne à l'intérieur d'un sub-patch pour contrôler son état d'occupation et activer ou désactiver le traitement du signal. Les objets *in*, *in ~*, *out* et *out ~* créent des entrées et des sorties spéciales de commande et de signal qui fonctionnent avec les entrées et les sorties de l'objet *poly ~*. La possibilité de désactiver le traitement audio à l'aide de messages **mute** fait de *poly ~* un outil précieux pour gérer la charge du CPU provoquée par les processus audio de en veille.