34-synthèse granulaire

Synthèse granulaire

Dans ce tutoriel, nous allons voir comment utiliser l'objet *poly ~* pour générer de grandes quantités de polyphonie afin de jouer le contenu d'un *buffer ~* de données d'échantillons. Nous allons exploiter la capacité de MSP à jouer des échantillons de données à partir du même *buffer ~* à de multiples vitesses arbitraires et points de temps pour explorer la technique de la ***synthèse granulaire***.

En termes simples, la synthèse granulaire est l'utilisation d'événements sonores très courts (ou parfois, moins courts) appelés "grains" pour générer des textures complexes. Bien que la littérature musicale et écrite sur la technique dépasse le cadre de ce didacticiel (voir Curtis Roads '***Microsound***' (MIT Press: 2004) pour une excellente exploration de ce sujet), nous en aborderons ici les bases. Alors que la synthèse granulaire classique repose sur l'utilisation de très petites quantités de données de tables d'ondes, la technique que nous allons explorer dans ce tutoriel utilise des données d'échantillons prises arbitrairement dans des fichiers son.

Dans le patcheur de notre didacticiel, nous allons créer un système de lecture algorithmique basé sur des valeurs aléatoires contraintes pour contrôler les paramètres suivants d'un moteur de lecture d'échantillons polyphoniques: taux, point de départ, durée, hauteur, amplitude. Nous verrons également comment l'ajustement des enveloppes modifie la sortie sonore.

Expérimenter avec le patcheur

Jetez un coup d'œil au patcheur du tutoriel. Il y a plusieurs boîtes de *nombres*, chacune contrôlant une partie de notre moteur de synthèse granulaire. La zone du patch étiquetée **1** est ***l'émetteur de grains*** proprement dit: un objet *metro* programme et envoie des messages **bang** à un objet *poly ~* qui a chargé 100 voix d'une abstraction nommée **polygrain ~**. La zone **2** nous permet de vérifier notre utilisation du CPU en fonction des paramètres de notre synthétiseur. Les zones **3** et **4** définissent les paramètres de synthèse - l'échantillon que nous utilisons, la zone à partir de laquelle extraire des grains et les paramètres du système de lecture des grains dans l'abstraction **polygrain ~**.

• Dans la zone **1** du patch, activez le son et montez le curseur de *gain ~*. En haut du patch, cliquez plusieurs fois sur l'objet *button* et écoutez le résultat. L'envoi d'un **bang** dans l'objet *poly ~* génère un seul "grain" d'audio. Activez l'objet *metro* en cliquant sur le *toggle* située en haut du patch.

L'objet *poly ~* de notre patcheur génère des grains: des salves uniques de lecture d'échantillons que nous pouvons contrôler dynamiquement en ajustant les paramètres. Les objets *metro* et *button* contrôlent l'émetteur de grains. A chaque fois que le *metro* se déclenche, il envoie un **bang** dans le *poly ~*, précédé du message de note, qui assigne le **bang** à la première voix disponible dans le *pol ~*. De plus, chaque **bang** de l'objet *metro* programme le suivant en ajustant la vitesse du *metro*. L'objet *random* génère une valeur aléatoire qui est ensuite envoyée à un objet *scale* avec une plage de sortie variable, définie par les paramètres **speedmin** et **speedmax** présents dans la zone de patch **4**.

Vérification du CPU

• Lorsque l'émetteur de grains est activé (c'est-à-dire que l'objet *metro* est configuré pour fonctionner), activez le *metro* dans la zone **2** du patcheur. La boîte de *nombres* située au bas de la logique de patcheur devrait afficher un nombre. Désactivez l'émetteur de grain en haut et regardez les résultats. Rallumez-le à nouveau.

L'objet *adstatus* nous permet de contrôler et de visualiser les aspects du pilote audio MSP en cours d'exécution. Tous les attributs visibles de la fenêtre ***Audio Status*** (disponible dans le menu **Max Options**) sont accessibles via l'objet *adstatus*. Le mode **cpu** de l'objet *adstatus* (défini par son argument) indique à l'objet de recevoir des messages **bang** et d'afficher l'utilisation actuelle du CPU par MSP. Remarquez que lorsque l'émetteur de grains est désactivé, l'utilisation du CPU tombe à **0**. Cela est dû au fait que notre abstraction *poly ~* se coupe elle-même lorsque sa lecture est terminée. Lorsqu'aucune note n'est émise, toutes les copies de l'abstraction *poly ~* devraient être muettes.

Réglage des paramètres

• Dans la zone de patch **3**, mettez en surbrillance une zone des objets *waveform ~* pour sélectionner une partie du *buffer ~* nommée **thegrain**. Remarquez que lorsque vous faites glisser l'un ou l'autre des objets *waveform ~*, les deux se mettent en surbrillance dans les mêmes régions. La sortie la plus à droite de l'objet *waveform ~* nous permet de les ***relier*** entre eux afin que vous puissiez utiliser plusieurs de ces objets pour travailler avec un *buffer ~* multicanal. Chargez un échantillon différent en utilisant les boîtes de *message* situées au-dessus de l'objet *buffer ~* et mettez en surbrillance différentes régions de l'échantillon. Les régions en surbrillance du *buffer ~* contrôlent l'endroit où l'émetteur de grain dessine ses données d'échantillon.

• Dans la zone de patch **4**, utilisez l'objet *preset* pour essayer différents paramètres pour notre émetteur de grains, puis essayez de saisir vos propres valeurs. Les boîtes de *nombres* du ***taux de grains*** contrôlent la plage de vitesse de *metro* dans la zone de patch **1**. Les ***durées des grains*** contrôlent la durée de chaque grain à l'intérieur du *poly ~*. Les valeurs de ***hauteur des grains*** fournissent une plage de vitesse de lecture des grains. Les contrôles ***d'amplitude des grains*** définissent la plage de volume de l'émetteur de grains et la ***pente des grains*** définit la netteté de l'attaque et du déclin de l'enveloppe de chaque grain. Remarquez comment différentes densités de grains modifient le son ainsi que l'utilisation du CPU des grains.

Avant d'examiner notre abstraction *poly ~*, remarquez l'effet de taux et de durées de grains plus longs et plus courts sur l'utilisation du CPU. Des durées de grain plus longues et des taux de grain plus courts ont pour conséquence qu'un plus grand nombre de voix à l'intérieur de *poly ~* soient actives à un moment donné - soit elles sont déclenchées plus fréquemment, soit elles prennent plus de temps pour se «libérer», soit les deux. Le résultat est une utilisation plus imoortante du CPU.

• Dans la zone de patcheur **1**, activez l'objet *toggle* attaché à la boîte de *message* intitulée **parallèle $ 1**. Redémarrez l'audio en éteignant et en rallumant le *dac ~*. Notez l'effet éventuel sur le CPU.

Selon l'architecture de votre ordinateur, vous pouvez tirer parti du multi-cœurs dans le processeur de votre ordinateur (ou de plusieurs processeurs si vous possédez une machine multiprocesseur) en divisant les ressources de l'objet *poly ~* sur plusieurs flux. En substance, cela permet de répartir les instances de l'objet *poly ~* sur les différents cœurs ou processeurs de votre ordinateur, permettant ainsi à des ensembles de voix de fonctionner en parallèle. En fonction de l'architecture du processeur de votre ordinateur, cela peut permettre d'améliorer considérablement les performances.

À l'intérieur du patch

• Double-cliquez sur l'objet *poly ~* pour afficher une instance de l'abstraction nommée **polygrain ~**. Jetez un coup d'oeil autour du patcheur.

L'abstraction *polygrain ~* reçoit un seul bang (via l'objet *in* en haut du patch) et l'utilise pour générer un grain d'audio, en utilisant la logique MSP en bas de l'abstraction. L'objet *trigger* en haut du patch établit clairement l'ordre des événements pour générer notre grain:

Tout d'abord, l'objet *thispoly* ~ reçoit un message **mute** 0 et 1 en succession immédiate. Cela ***active*** (débloque) le traitement du signal dans l'instance et lui donne l'état "busy" de sorte qu'il ne recevra plus de messages de note jusqu'à ce que le grain soit terminé.

Ensuite, un **bang** est envoyé pour générer une ***amplitude*** aléatoire pour le grain, qui est placée dans le côté droit de l'objet * ~ intitulé "How loud is this grain?". Cet objet * ~ contrôle la mise à l'échelle de la sortie de l'objet *line* ~ ci-dessus, qui définit l'enveloppe de grain.

Troisièmement, une ***hauteur*** de son aléatoire est sélectionnée et transformée en un multiplicateur de durée pour les objets *line* ~ contrôlant la lecture de l'échantillon et son enveloppe d'amplitude. L'objet *! /* divise la hauteur entrante en **1.**, de sorte qu'une hauteur demandée de **2.** indique aux objets situés en aval de multiplier leurs durées par **0,5** (moitié moins longue, et une octave de plus).

Quatrièmement, une ***durée*** aléatoire est générée, qui configure les paramètres des objets *line* ~ afin qu'ils génèrent les valeurs de mise à l'échelle et de décalage appropriées pour la longueur du grain.

Enfin, un grain est déclenché en générant un point de départ aléatoire basé sur les zones mises en évidence dans l'objet *waveform* ~ du patch principal. Ce **bang** génère finalement deux messages qui commandent aux deux objets *line* ~ de générer la courbe de lecture pour l'objet *play* ~ et l'enveloppe d'amplitude pour les objets * ~.

Une fois que l'objet *line* ~ «enveloppe» a terminé, il envoie un **bang** pour couper et la mettre en «free» (**0**) pour pouvoir recevoir un nouveau message.

• Dans le menu **Fichier** de Max, sélectionnez ***Modifier en lecture seule***. Cela vous permettra de déverrouiller la copie de l'abstraction **simplegrain ~** que vous visualisez. Déverrouillez le patch et placez des «points de surveillance» sur certains cordons de connexion pour contrôler leurs valeurs. Dans la fenêtre ***Watchpoints***, vous devriez voir comment les différentes valeurs des paramètres de grain du patcheur principal se traduisent en valeurs pour l'algorithme de synthèse utilisé ici.

Résumé

L'objet *poly* ~ vous permet d'avoir un grand nombre d'instances d'un patcheur MSP simple et unique. Vous pouvez utiliser *send* et *receive* pour communiquer avec toutes les instances d'une abstraction *poly* ~, qui peuvent être distribuées sur plusieurs cœurs ou processeurs avec le message *parallel*. L'objet *adstatus* vous permet d'accéder à certains aspects du pilote audio MSP et de les modifier; l'argument **cpu** de l'objet vous permet quelle part du CPU de votre ordinateur vous utilisez avec un patch.