8-routage des signaux

introduction

Dans ce didacticiel, nous allons examiner les différentes façons d'acheminer les connexions audio de MSP, ainsi que l'envoi de messages à distance dans Max, en mettant l'accent sur l'utilité de cette méthode pour la conception d'un réseau de signaux. En cours de route, nous verrons comment le gain peut être exprimé en décibels et comment fonctionne l'objet *phasor ~*.

Connexions de signal à distance: *send ~* et *receive ~*

Les cordons de raccordement qui connectent les objets MSP sont différents des cordons de connections normaux, car ils agissent différemment. Ils décrivent l'ordre des calculs dans un réseau de signaux. Ces objets connectés seront utilisés pour calculer un bloc entier d'échantillons pour la prochaine portion de son émise par votre ordinateur.

Les objets Max peuvent communiquer à distance sans cordons de connections en utilisant des objets *send* et de *receive* (et certains objets similaires tels que *value* et *pv*). Vous pouvez également utiliser des points-virgules dans les boîtes de *message* pour transmettre des valeurs aux objets *receive* par leur nom. Vous pouvez également transmettre des signaux MSP à distance. Cependant, plutôt que d'utiliser des objets normaux de *send* et de *receive*, deux objets MSP existent spécifiquement pour la transmission de signaux à distance: *send ~* et *receive ~*.

Les deux objets *send ~* et *receive ~* fonctionnent de manière très similaire à *send* et *receive*, mais sont uniquement destinés à être utilisés avec des signaux MSP. Max vous permettra de connecter des cordons de connections normaux à *send~* et *receive~* , mais seuls des signaux seront transmis via *send~* vers le *receive ~* correspondant.

Il existe quelques différences importantes entre les objets Max *send* et *receive* et les objets MSP *send ~* et *receive ~*.

1- Les noms de *send* et de *receive* peuvent être abrégés en *s* et *r*; les noms de *send ~* et de *receive ~* ne peuvent pas être abrégés de la même manière.

2- Un message Max peut être envoyé à un objet *receive* à partir de plusieurs objets autres que *send*, comme *float*, *forward*, *grab*, *if*, *int* et *message*; *receive ~* peut recevoir un signal uniquement que d'un objet *send ~* qui partage le même nom.

3- Si *receive* n'a pas d'argument dactylographié, il dispose d'une entrée permettant de recevoir les messages **set** afin de définir ou de modifier son nom; *receive ~* dispose également d'une entrée à cet effet, mais il est néanmoins nécessaire d'avoir un argument dactylographié pour lui donner un nom de destination initial.

4- L'objet Max *send*, une fois créé, ne peut pas changer de destination (c'est ce que fait l'objet Max *forward*.) L'objet MSP *send ~* peut changer de destination avec un message **set**.

Des exemples de chacune de ces utilisations sont disponibles dans le patch du didacticiel.

Acheminement d'un signal: *gate ~*

L'objet MSP *gate ~* fonctionne de manière très similaire à l'objet Max *gate*. Tout comme *gate* est utilisé pour diriger des messages vers une destinations parmi d'autres ou pour couper complètement

le flux de messages, *gate ~* dirige un signal vers différents endroits ou le coupe du reste du réseau de signaux.

Dans l'exemple de patch, les objets *gate ~* sont utilisés pour router les signaux vers la sortie audio gauche, la sortie audio droite, les deux ou aucune, en fonction du numéro reçu de l'objet *umenu*.

Il convient de noter que le fait de changer la sortie choisie d'une *gate ~* pendant le passage d'un signal audio peut provoquer un clic audible car le signal passe brusquement d'une sortie à l'autre. Pour éviter cela, vous devez généralement concevoir votre patch de manière à ce que la sortie de l'objet *gate ~* ne soit modifiée que lorsque le signal audio qui le traverse est à zéro ou lorsque l'audio est désactivé. (Aucune précaution de ce type n'a été prise dans ce patch du didacticiel.)

Les sorties fermées de *gate ~* ne cessent pas d'envoyer des signaux, elles continuent d'envoyer un signal avec une valeur constante de 0. Si l'idée d'un signal constant vous semble étrange, imaginez-le à 44 100 secondes par seconde.

Interférence d'onde

C'est un fait physique fondamental que, lorsque nous additionnons deux ondes sinusoïdales de fréquences différentes, nous créons des interférences entre les deux ondes. Comme elles ont des fréquences différentes, elles ne seront généralement pas exactement en phase l'une avec l'autre; à certains moments, elles ne seront généralement pas en phase pour s'additionner de manière constructive, mais à d'autres moments, elles s'additionneront de manière destructive, s'annulant mutuellement dans une certaine mesure. Elles n'arrivent précisément en phase l'une avec l'autre qu'à un rythme égal à la différence de leurs fréquences. Par exemple, une sinusoïde à 1000 Hz et une autre à 1002 Hz entrent en phase exactement 2 fois par seconde. Dans ce cas, leur fréquence est suffisamment proche pour que nous ne les entendions pas comme deux sons distincts. Au lieu de cela, nous entendons leur modèle récurrent d'interférence constructive et destructrice sous forme de battements se produisant à une fréquence inférieure à 2 Hz, une fréquence connue sous le nom de ***fréquence de différence*** ou ***fréquence de battement***. (fait intéressant, nous entendons les deux ondes comme un seul son avec une fréquence de battement subaudio inférieure à 2 Hz et une fréquence audio de 1001 Hz.)

Lorsque le patch de l'exemple est ouvert, un objet *loadbang* envoie les valeurs de fréquence initiales aux deux objets *cycle ~* du patch en les réglant sur 1000 Hz et 1002 Hz; nous nous attendons donc à ce que ces deux sons émis ensemble provoquent une fréquence de battement de 2 Hz. Il envoie également des valeurs initiales aux objets *umenu* qui, à leur tour, définissent les objets *gate ~*, dirigeant un son vers la sortie audio gauche et un autre vers la sortie audio droite. Une quatrième valeur appelée **Depth** est également définie. Tout ceci est accompli en utilisant une boîte de *message* pour régler à distance les valeurs captées par des objets Max *receive* dans le patcheur.

• Dans le didacticiel, cliquez sur *ezdac ~* pour activer l'audio, puis utilisez le *slider* intitulé **Volume** pour régler l'intensité du son à un niveau confortable. Notez que les battements se produisent exactement deux fois par seconde. Essayez de changer la fréquence de l'oscillateur B en utilisant divers autres nombres proches de 1 000 (en utilisant la boîte de *nombre* étiquetée **fréquence**) et notez l'effet. Lorsque la fréquence de différence se rapproche d'une fréquence audio (par exemple, dans la plage de 20-30 Hz), vous ne pouvez plus distinguer les battements individuels et l'effet devient davantage un changement de timbre. Augmentez encore la différence et vous commencez à entendre deux fréquences distinctes.

• ***Tangente philosophique***: on peut démontrer mathématiquement et empiriquement que, lorsque deux sons sinusoïdaux sont additionnés, leur schéma d'interférence se répète à une vitesse égale à la

différence de leurs fréquences. Cela explique apparemment pourquoi nous entendons des battements; l'amplitude varie manifestement au rythme de la différence. Cependant, si vous écoutez ce patch avec des écouteurs de sorte que les deux sons n'aient jamais l'occasion d'interférer mathématiquement, électriquement ou dans l'air, vous entendez toujours les battements! Ce phénomène, connu sous le nom de ***battement binaural***, est dû à des "interférences" qui se produisent dans le système nerveux. Bien que cette interférence soit d'une nature physique très différente de celle des ondes sonores dans l'air, nous les ressentons comme similaires. Une expérience comme celle-ci démontre que notre système auditif façonne activement le monde que nous entendons.

Amplitude et amplitude relative

Le *slider* intitulé «Volume» a été configuré pour avoir une plage de 101 valeurs, allant de **0** à **100**, ce qui facilite la conversion de sa sortie en une valeur flottante allant de **0** à **1** en divisant simplement par 100. (Le point décimal dans l'argument tapé dans l'objet / garantit une division en nombre flottant).

Les objets * ~ utilisent la valeur d'amplitude spécifiée pour mettre à l'échelle le signal audio avant qu'il ne soit envoyé à l'*ezdac~*. Si les deux oscillateurs sont envoyés vers la même entrée de l'*ezdac ~*, leur amplitude combinée sera égale à 2. Il est donc prudent de maintenir le facteur d'échelle d'amplitude à 0,5 ou moins. Pour cette raison, la valeur d'amplitude que l'utilisateur pense être comprise entre 0 et 1 est en réalité maintenue entre 0 et 0,5 par l'objet *** 0.5** situé sous le *slider*.

En raison de la large gamme d'amplitudes audibles possibles, il peut parfois être plus judicieux d'afficher le volume numériquement en termes d'échelle logarithmique de décibels (dB) plutôt qu'en termes d'amplitude absolue. L'échelle des décibels fait référence à l'amplitude ***relative***, c'est-à-dire l'amplitude d'un signal par rapport à une certaine amplitude de référence. La formule de calcul de l'amplitude en décibels est la suivante:

$$dB = 20 (log10 (A / Aref))$$

où A est l'amplitude mesurée et Aref est une amplitude de référence fixe.

L'objet *AtodB* utilise une amplitude de référence de 1 dans la formule indiquée ci-dessus et convertit l'amplitude en dB. Si vous désactivez l'audio (pour des raisons de sécurité) et si vous augmentez la valeur du *slider* (assurant une sortie de 1), vous verrez la sortie en décibels de l'objet *AtodB* afficher 0 (c'est-à-dire un gain unitaire). Chaque division par deux de l'amplitude du curseur est approximativement égale à une réduction de 6 dB en décibels.

Valeur de signal constante : *sig ~*

La plupart des réseaux de signaux nécessitent certaines valeurs changeantes (comme une enveloppe d'amplitude pour faire varier l'amplitude dans le temps) et certaines valeurs constantes (comme une valeur de fréquence pour maintenir un oscillateur à une hauteur constante). En général, on fournit une valeur constante à un objet MSP sous la forme d'un message **flottant**, comme nous l'avons fait dans ces exemples en envoyant une fréquence dans l'entrée gauche d'un objet *cycle ~*.

Cependant, dans certains cas, on souhaite combiner des valeurs constantes et des valeurs variables dans la même entrée d'un objet MSP. Les entrées qui acceptent soit un **float**, soit un **signal** (comme l'entrée gauche du *cycle ~*) ne parviennent pas à combiner les deux. En général, les objets MSP ***ignoreront*** une valeur à virgule flottante par déférence pour une entrée de signal. Par conséquent, il est nécessaire d'avoir un objet qui émet une valeur constante comme signal si vous avez besoin de

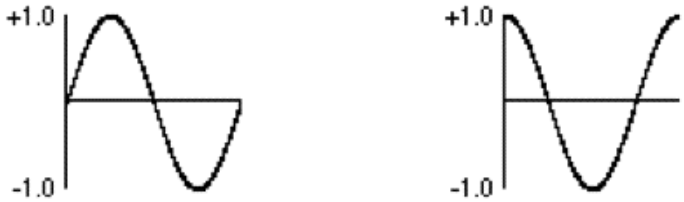
combiner une valeur fixe et une valeur variable dans une chaîne de signaux MSP.

Une façon de combiner un message numérique Max (un **int** ou un **float**) avec un signal est de convertir le nombre en un signal constant avec l'objet *sig* ~. La sortie de *sig* ~ est un signal avec une valeur constante, déterminée par le nombre reçu dans son entrée.

Dans l'exemple de patch, **l'Oscillateur B** combine une fréquence constante (fournie sous forme de valeur **float** vers *sig* ~) à un décalage de fréquence variable (les valeurs de signal provenant d'un objet *receive* ~ appelé FreqB +). La somme de ces signaux sera la fréquence de l'oscillateur à un instant donné.

Changer la phase d'une forme d'onde: *phasor* ~

Dans la plupart des cas, le déphasage d'une onde audio isolée n'a pas d'effet sensible sur la perception. Par exemple, une onde sinusoïdale dans la gamme audio sonne exactement comme une onde cosinusoïdale, même s'il existe une différence de phase théorique d'un quart de cycle. Pour cette raison, jusqu'à présent, nous ne nous sommes pas préoccupés jusqu'à présent de l'entrée de phase la plus à droite de l'objet *cycle* ~.



Une onde sinusoïdale compensée par un quart de cycle est une onde cosinusoïdale

Cependant, il existe des raisons très utiles pour contrôler le décalage de phase d'une onde. Par exemple, en laissant la fréquence du *cycle* ~ à **0** et en augmentant continuellement son déphasage, vous pouvez modifier sa valeur instantanée (comme si sa fréquence était positive). Le décalage de phase d'une sinusoïde est généralement indiqué en degrés (un cycle complet vaut 360 °) ou en ***radians*** (un cycle complet a une longueur de 2 * π radians). Dans l'objet *cycle* ~, la phase est exprimée en cycles d'ondes; ainsi, un décalage de π radians correspond à la moitié d'un cycle, soit **0.5**. En d'autres termes, lorsque la phase varie de 0 à 2π radians, elle varie de **0** à **1** cycle d'onde. Cette façon de décrire la phase est pratique car elle nous permet d'utiliser la plage de signaux commune de 0 à 1.

Ainsi, si nous faisons varier le déphasage d'un objet *cycle* ~ stationnaire (0 Hz) de façon continue de 0 à 1 au cours d'une seconde, la sortie résultante est une onde cosinusoïdale de fréquence 1 Hz.

L'objet *phasor* ~ est un générateur de signal MSP très utile qui produit simplement une rampe de **0** à **1** à une fréquence donnée. Comme pour l'objet *cycle* ~, sa fréquence peut être définie par un argument à l'objet, une valeur à virgule flottante dans son entrée de gauche ou un signal. Nous entendrons plus loin dans notre didacticiel comment cela sonne (c'est une onde en dents de scie et pas particulièrement agréable), mais pour l'instant, utilisons-le pour changer la phase (d'où *phasor*~) de l'objet *cycle* ~ à gauche du patcheur du tutoriel.

• Cliquez sur la boîte de *message* située au-dessus des objets *line* ~ dans la partie inférieure gauche du patch du didacticiel. Vous devriez entendre l'une des ondes sinusoïdales commencer à baisser et à monter lentement en hauteur, ce qui fait que le battement entre les deux sinus connectés à la sortie audio change au fil du temps.

Le *cycle* ~ en bas à gauche du patch ***module*** l'objet *cycle* ~ en haut à droite en ajoutant une valeur à sa fréquence actuelle (constante), définie par l'objet *sig* ~.

> **Note historique**: À l'époque de la synthèse MusicN, le ***phasor*** était le seul ***générateur d'unités*** capable de produire un son continu. Un *phasor* devait être connecté à une table d'ondes pour générer d'autres formes d'onde. Dans MSP, le *cycle* ~, le *triangle* ~ et les autres oscillateurs ont des *phasors* internes, mais le concept de *phasor* est si utile pour de nombreuses opérations que nous en avons fourni une version autonome.

Reception d'un signal différent

La partie restante du patch du didacticiel existe simplement pour démontrer l'utilisation du message set à l'objet *receive* ~. Il s'agit d'une autre façon de modifier le flux de signaux dans un réseau. Comme pour l'objet *send* ~, vous pouvez modifier le nom de l'objet *receive* ~ avec un message set, en lui demandant d'obtenir son entrée à partir d'un autre objet (ou d'autres objets) *send* ~.

• Cliquez sur la boîte de *message* contenant **set sawtooth**. Les deux objets *receive* ~ connectés reçoivent maintenant leur signal de l'objet *phasor* ~ situé dans le coin inférieur droit de la fenêtre. Cliquez sur les boîtes de *message* contenant les messages **set outL** et **set outR** pour recevoir à nouveau les sons sinusoïdaux. Avec la destination dents de scie sélectionnée, modifiez la fréquence de *phasor* ~ dans le coin inférieur droit du patcheur. Remarquez que lorsqu'il est réglé sur une fréquence audible, la rampe de l'objet *phasor* ~, le fait sonner comme un oscillateur riche en harmoniques. Il s'agit d'une onde en dents de scie, une forme d'onde couramment utilisée dans la conception de synthétiseurs. Comme l'objet *phasor* ~ ne fait que produire des rampes de **0** à **1** (et non de **-1** à **1**), utiliser un *phasor* ~ comme source audio n'est pas le meilleur moyen d'obtenir ce son: l'onde *saw* ~ de MSP fera plutôt l'affaire.

Résumé

Il est possible d'établir des connexions de signaux sans cordons de connexion en utilisant les objets *send* ~ et *receive* ~ de MSP, qui sont similaires aux objets *send* et *receive* de Max. Le message **set** peut être utilisé pour changer le nom d'un objet *send* ~ ou *receive* ~, et ainsi changer la façon dont les signaux sont acheminés. Le flux de signaux peut être acheminé vers différentes destinations au sein d'un patcheur, ou entièrement coupé, à l'aide de l'objet *gate* ~, qui est l'équivalent MSP de l'objet *gate* de Max.

L'objet *cycle* ~ peut être utilisé non seulement pour les ondes audio périodiques, mais aussi pour les fonctions de commande sub-audio: vous pouvez lire la forme d'onde d'un objet *cycle* ~ à n'importe quelle vitesse en maintenant sa fréquence à 0 Hz et en modifiant sa phase continuellement de 0 à 1. L'objet *phasor* ~ est approprié pour changer la phase d'une forme d'onde *cycle*~ de cette façon; vous pouvez aussi utiliser un objet l*ine* ~ pour créer une rampe qui passe par la table d'onde de l'objet.

L'objet *sig* ~ convertit un nombre en un signal constant; il reçoit un nombre dans son entrée et émet un signal de cette valeur. Cet objet est utile pour combiner des valeurs constantes avec des signaux variables. Le mélange des sons de fréquences légèrement différentes crée des interférences entre les ondes, ce qui peut créer des battements et d'autres effets de timbre.