

Introduction à max/msp

Option électronique et signal pour la musique - ensea
S. Reynal

26 janvier 2022

1 Introduction

Ce TP vous propose de vous initier à MAX/MSP à travers la réalisation d'un embryon de synthétiseur de type "synthèse soustractive". MAX/MSP est un environnement de programmation graphique (i.e., par icônes ou "blocs" que l'on relie par des fils) dédié à l'audio numérique, qui permet de réaliser les fonctions essentielles d'un système de traitement audio :

- acquisition et restitution d'un signal via une interface audio ;
- traitements audio (oscillateurs, mélangeurs, filtres, non-linéarités, vca et vcf, fonctions mathématiques, FFT,...) ;
- contrôle et interface homme-machine (interface graphique, messages MIDI en provenance ou à destination d'un instrument de musique externe, envoi et réception de données sur le réseau, ...).

Max/MSP a été conçu par l'IRCAM et est distribué par la société européenne Ableton située à Berlin. Un exemple de patch audio est illustré figure 1.

MAX/MSP vs. puredata~ MAX/MSP est très proche, dans l'esprit comme dans la syntaxe, de l'environnement `puredata~`. Les deux environnements se sont d'ailleurs mutuellement inspirés l'un de l'autre. Contrairement à MAX/MSP, `puredata` est toutefois gratuit et opensource, mais possède une interface graphique moins riche que celle de MAX/MSP (cf. figure 2). MAX/MSP est également livré avec des bibliothèques additionnelles très puissantes (ainsi `jitter`, qui permet d'implémenter du traitement d'image/video). Mais connaître `puredata~` permet de se servir de MAX/MSP sans véritable effort supplémentaire, et réciproquement.

Utilisation `puredata~` comme MAX/MSP représentent des outils puissants et rapides de prototypage d'applications, algorithmes et interfaces graphiques pour les systèmes de traitement audio numériques (avant implémentation sur système embarqué par exemple), et c'est à ce titre que nous vous

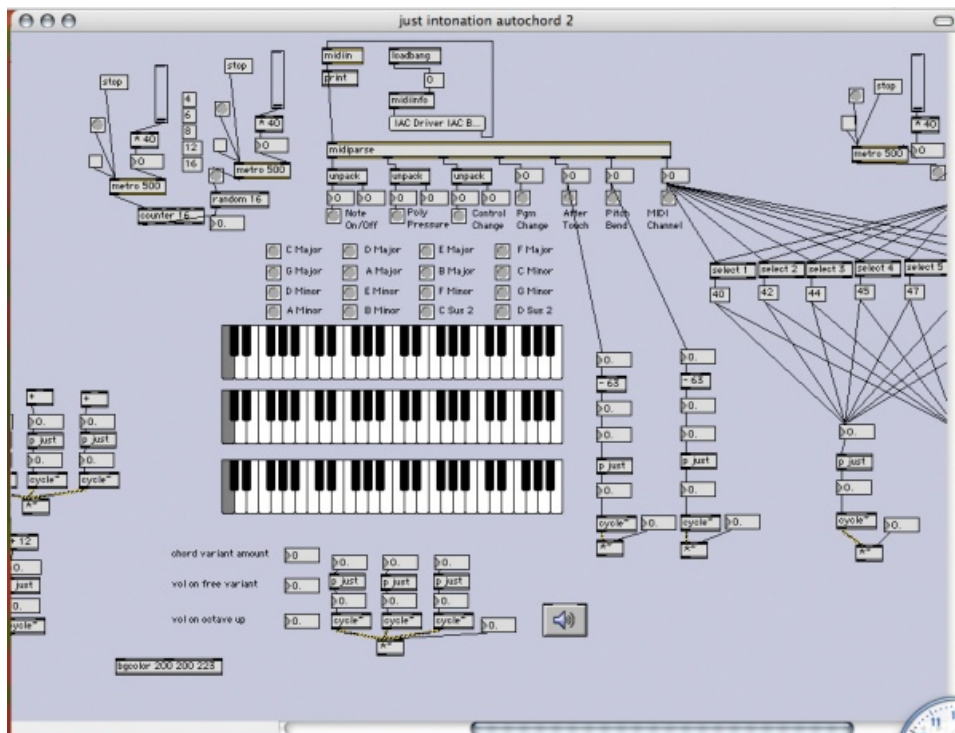


FIGURE 1 – Un patch typique sous max/msp. Chaque boîte rectangulaire réalise une fonction. Chaque fonction peut prendre des arguments d’initialisation (ou arguments ”par défaut”).

proposons de vous y initier aujourd’hui. Ils complètent plus qu’ils ne remplacent Matlab, java ou C ; il est d’ailleurs facile de les interfacer avec ces deux derniers langages.

2 Installation

Vérifier tout d’abord si MAX/MSP est bien installé sur votre machine. Sans licence payante, vous pouvez utiliser le logiciel pendant un mois (licence ”demo”), avec les mêmes possibilités qu’avec la licence payante.

Créer un nouveau ”patch” vide (File -> New Patcher) et l’enregistrer sous ”synth-soustractif”.

Quelques rudiments

- On bascule du mode Edit au mode Execution par ”View -> Edit”. Vérifier quel raccourci clavier y est associé, car on son temps à basculer entre ces deux modes.
- La barre d’outils supérieure permet d’insérer des boîtes dans le patch : Object (réalise une fonction), Message (permet d’envoyer un message

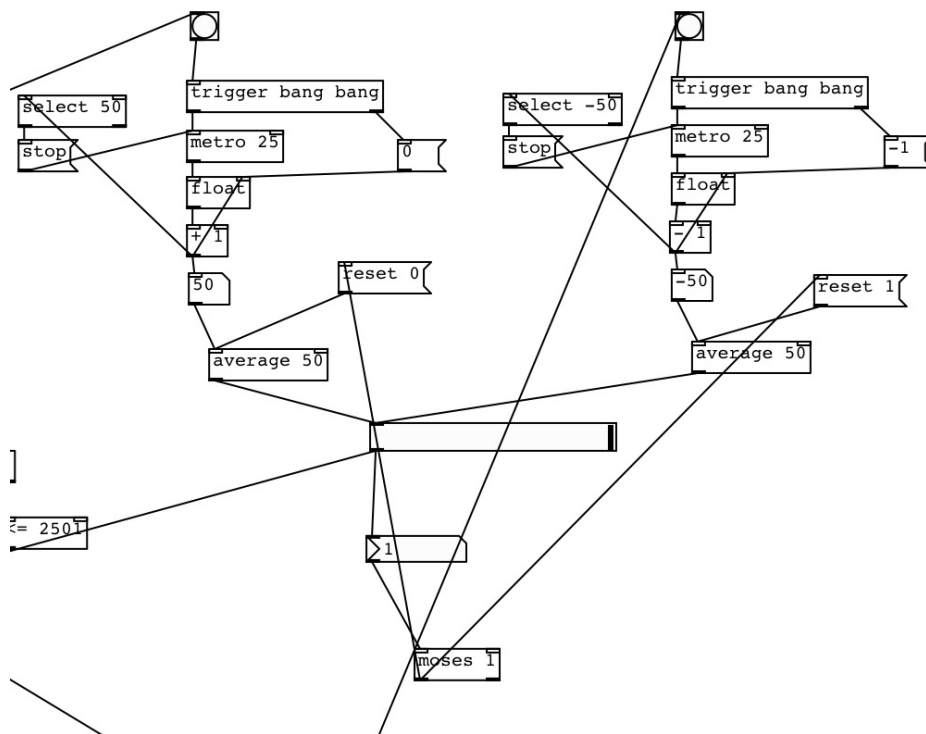


FIGURE 2 – Un exemple de patch sous puredata.

quelconque à un objet en cliquant dessus) et Numbers (message particulier à valeur numérique et éditable facilement à la souris) sont les briques essentielles. Une fois une boîte insérée, il suffit d'entrer le nom de la fonction souhaitée à l'intérieur du rectangle, éventuellement suivi de ses paramètres par défaut. Si la fonction n'est pas reconnue par MAX/MSP, le rectangle est cerclé d'orangé. Il reste enfin à connecter les *inlets* (entrées, en haut) et *outlets* (sorties, en bas) aux autres blocs du patch.

- dans les menus "buttons" et "Max for Live objects" de la barre d'outil se trouvent les composants de base de l'interface graphique.
- Il existe deux types de fils de connexion : les gris, qui transportent des messages de contrôle, et les jaune/gris, qui transportent des signaux audio.
- Tous les objets dont le nom se termine par un `~` produisent (et souvent admettent en entrée) des signaux audio (par exemple, `phasor~` est un oscillateur audio et `lores~` est un filtre passe-bas) ; les autres produisent des messages de contrôle (par exemple `line` produit une rampe de contrôle et `print` permet d'imprimer sur la console).
- Pour pouvoir entendre quoi que ce soit produit par MAX/MSP, il faut activer le processing audio en cliquant sur le bouton situé dans le coin inférieur droit.
- L'objet `print` permet d'afficher tout message connecté sur son entrée, dans la Max/Console (bouton situé dans la barre d'outils de droite)
- Pour obtenir une **aide contextuelle** sur un objet, faire "clic droit" sur sa boîte. Il existe également une aide globale via le menu Help.

3 Synthèse soustractive

Le son synthétique basé sur la forme d'onde "en dent de scie" (sawtooth) filtrée par un filtre passe-bas est probablement le son le plus emblématique de la musique électronique depuis ses débuts dans les 60's jusqu'à aujourd'hui. Des centaines de synthétiseurs ont été construits sur ce principe, dont les fameux Roland Juno ou les Moog Voyagers. Le patch que nous allons construire aujourd'hui est toutefois une émulation extrêmement rudimentaire de ces synthétiseurs, et constitue davantage un prétexte pour découvrir MAX/MSP et ses fonctionnalités de base.

4 Réalisation du patch

4.1 Deux oscillateurs en dent de scie (figure 3)

Notre brique de base sera un mélange de deux oscillateurs en dent de scie légèrement désaccordés. L'objet `phasor~` génère un signal entre 0 et 1,

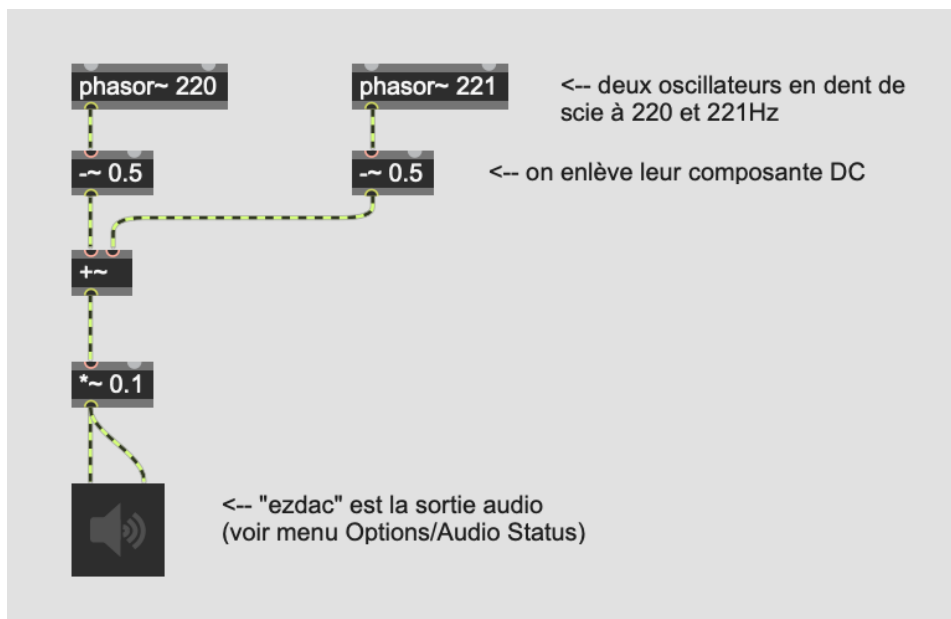


FIGURE 3 – Premier patch : l'oscillateur...

et le second bloc permet de le recentrer autour de 0. L'objet `+~` réalise la sommation. La sortie est divisée par 10 pour éviter de saturer la sortie audio du PC (par convention, l'amplitude maximale du signal envoyé sur l'objet `dac~` est 1.0 et au-delà la sortie audio du PC est saturée).

A faire :

- réaliser le patch et écouter
- jouer sur la fréquence de l'oscillateur de droite et vérifier l'effet sur l'épaisseur du son.

4.2 Filtrage soustractif (figure 4)

On insère un filtre passe-bas `lores~` afin de filtrer certaines harmoniques. Le premier *inlet* de `lores~` admet le signal à filtrer, le second *inlet* (ou de manière équivalente, le premier argument d'initialisation, "660"), la fréquence centrale, et le troisième *inlet* (ou le second argument, "0.99"), le facteur de qualité. Pour l'instant la fréquence du filtre est fixe, mais nous verrons un peu plus loin comment la rendre dynamique en injectant un message de contrôle sur l'*inlet* 2.

A faire :

- jouer sur la fréquence centrale du filtre et le facteur de qualité.

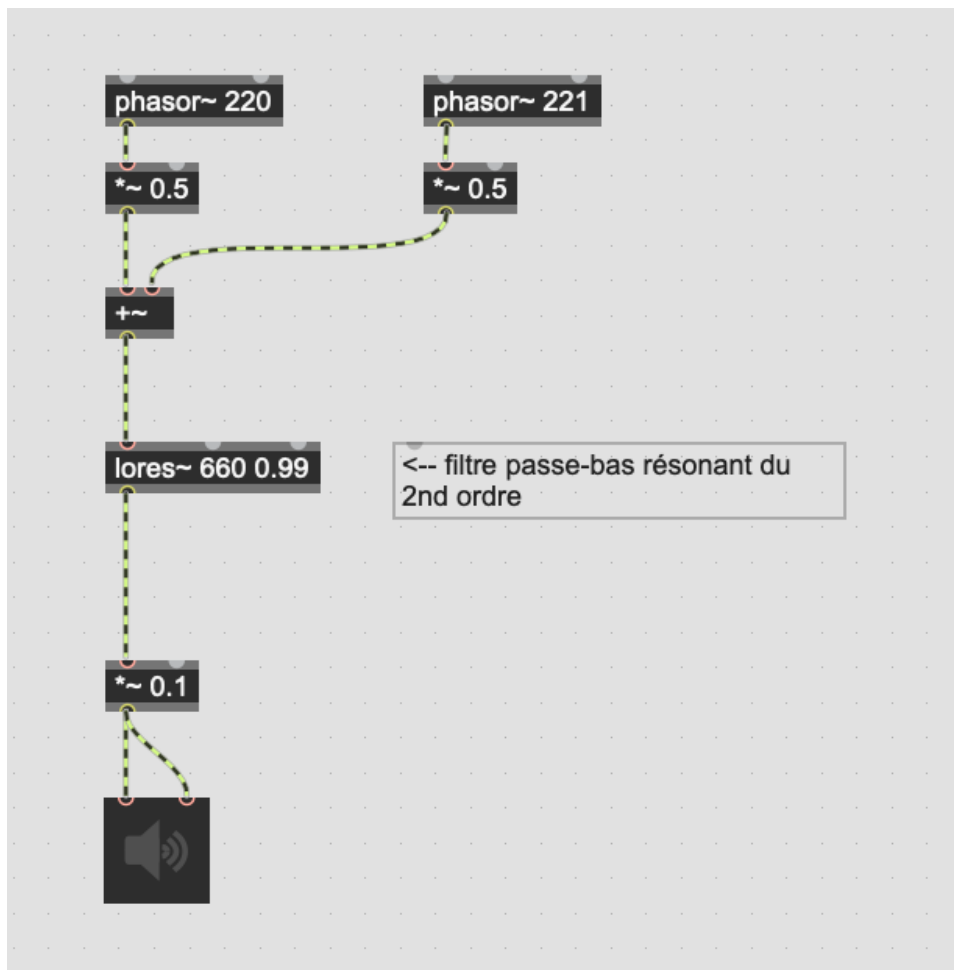


FIGURE 4 – Ajout d'un filtre passe-bas.

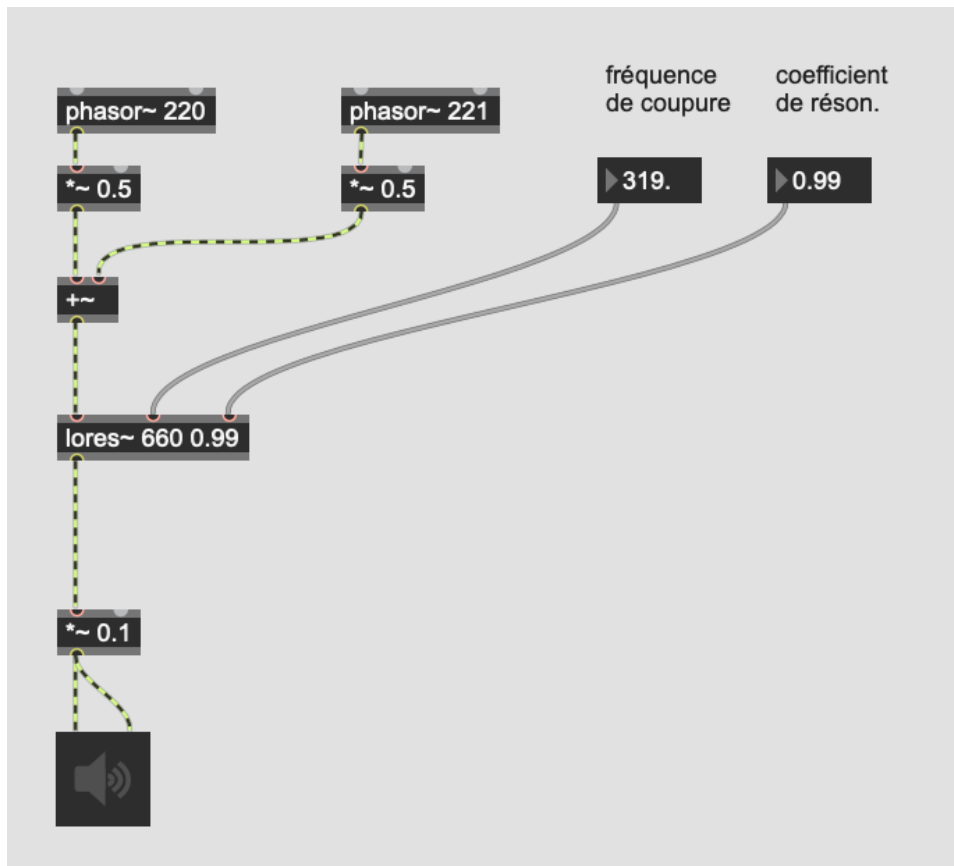


FIGURE 5 – Contrôle du filtre et de la résonance.

4.3 Contrôle statique du filtre (figure 5)

On souhaite contrôler facilement la fréquence du filtre. Pour cela ajouter un objet "numbers (f)" (=float) que l'on connectera à l'inlet 2 de `lores~`. Faire de même pour le contrôle du niveau de sortie.

4.4 Contrôle dynamique du filtre (figure 6)

Souvent la personnalité particulière d'un son synthétique comme d'un son naturel vient de la façon dont son spectre évolue dynamiquement. Par exemple, une corde de piano ou de guitare "perd" des harmoniques par dissipation au cours du temps (en quelques secondes en général), et le son d'une trompette passe par un pic de brillance peu après l'attaque par les lèvres.

Ici l'idée est de modifier dynamiquement la fréquence du filtre avec un générateur d'enveloppe rudimentaire de type `attack/decay` (enveloppe en triangle). Pour cela, on va provisoirement déconnecter le slider de contrôle du filtre, et le remplacer par une connexion à l'outlet d'un objet `line`. Ce dernier

objet produit en sortie une valeur numérique évoluant d'une valeur à une autre en un temps déterminé et ce de manière affine. `line` accepte en entrée un message composé d'une paire de nombres : le premier représente la valeur cible à atteindre, le second le temps (en ms) nécessaire pour l'atteindre.

A faire :

- Dans un premier temps, réaliser le sous-patch proposé figure 6 dans un coin libre de votre patch, et le tester. Les deux boîtes du haut sont des objets de type "message" (menu "put -> message") : en mode "exécution", un clic sur la boîte envoie le message correspondant à l'objet `line`. Un clic sur le message "1 1500" déclenche l'attaque : la sortie de `line` passe alors de sa valeur actuelle (zéro par défaut) à la valeur 1 en 1500 ms. Le second message "0 2000" correspond à la phase "decay" lors de laquelle le filtre se refermera. L'objet `print` permet de visualiser l'évolution de la sortie de l'objet `line` dans la console. La valeur "fc=" affichée correspond à la sortie de `line` multipliée par 1000 et décalée de 20, soit une fréquence comprise entre 20Hz et 1020Hz.
- quand c'est validé, connecter la sortie de l'objet `+ 20` au second inlet du filtre `bp~`. Tester en cliquant successivement sur les deux messages d'entrée de `line`.

4.5 Générateur d'enveloppe automatique (figure 7)

On veut automatiser le générateur d'enveloppe, et ne pas avoir besoin d'appuyer successivement sur les deux messages. Pour cela il faut envoyer les deux messages séquentiellement, le second (decay) devant être émis lorsque la phase d'attaque arrive à son terme. On utilise plusieurs ingrédients :

- on va construire dynamiquement le message "1 attack-time" et "0 decay-time" en utilisant la notation `$1` qui représente le message présent sur l'inlet 1.
- on va devoir temporairement stocker les valeurs des sliders : ce sera fait grâce aux objets `float` qui stockent une valeur présente sur l'inlet 2 jusqu'à ce qu'un message "bang" soit envoyé sur l'inlet 1 et déclenche la sortie de la valeur stockée sur l'outlet.
- l'objet `trigger` évalue ses arguments de droite à gauche : ici il permet d'envoyer des messages "bang" aux boîtes suivantes sur un clic du bouton, mais dans un ordre bien défini (la boîte `delay` PUIS la boîte `float` de gauche).
- enfin l'objet `delay` retarde le message entrant d'une durée spécifiée sur son inlet 2 : ici cela permet de déclencher la phase de decay uniquement lorsque la phase d'attaque est terminée.

A faire :

- tester le sub-patch ;

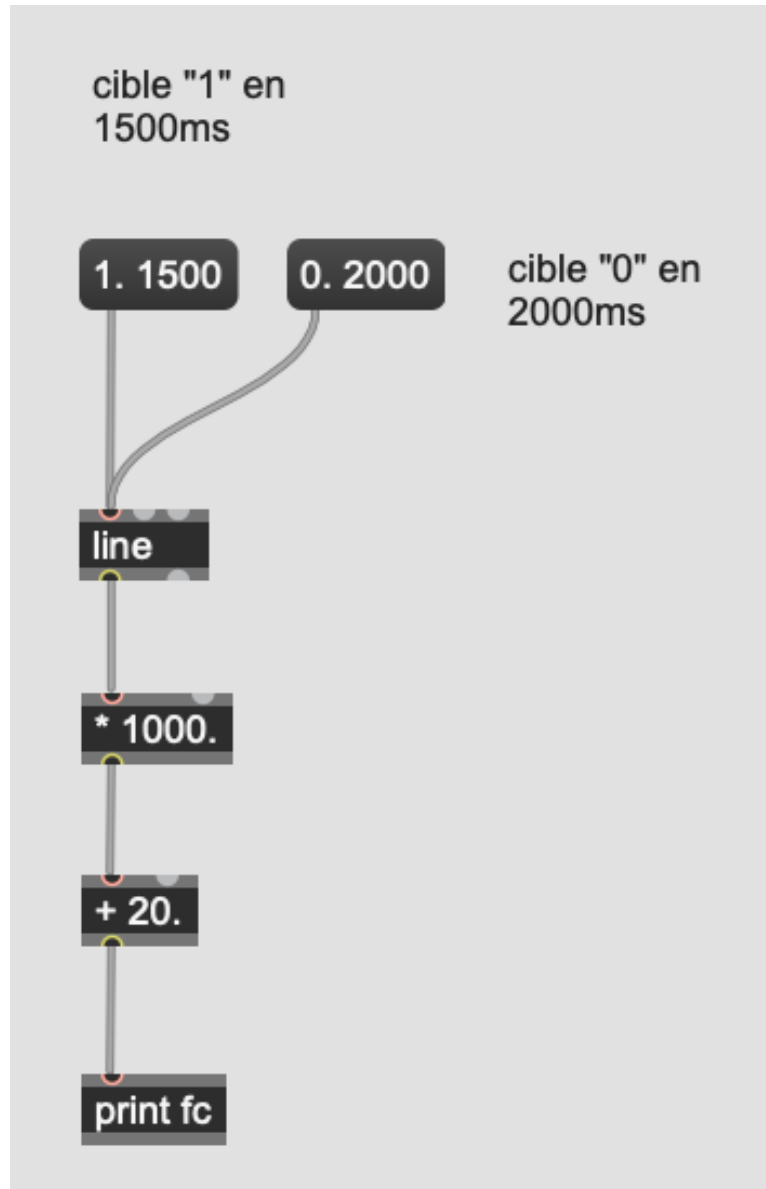


FIGURE 6 – Enveloppe attack/decay pour le contrôle dynamique du filtre.

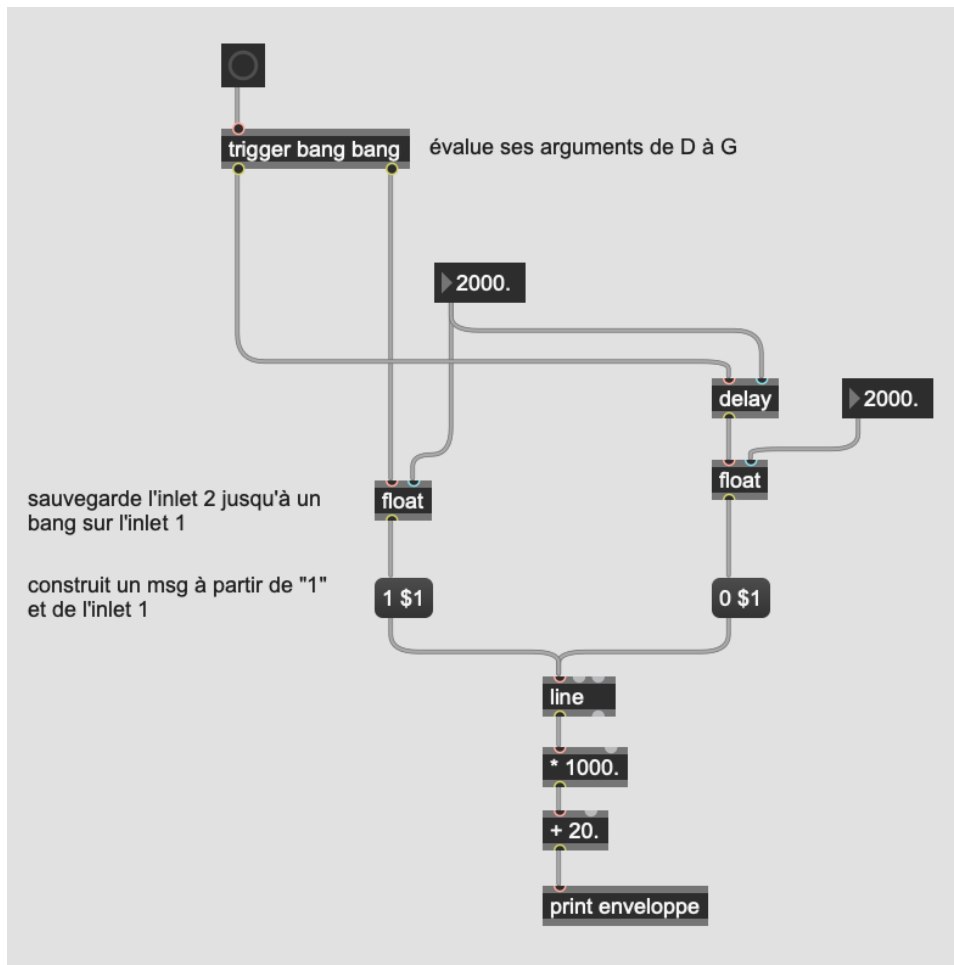


FIGURE 7 – Générateur d'enveloppe automatique.

- une fois validé, le connecter au reste de votre patch actuel en effaçant ce qui est superflu.

4.6 Subpatch du générateur d'enveloppe (figure 8)

Pour rendre les patches plus lisibles on utilise généralement des subpatches : ici on va créer un subpatch pour le générateur d'enveloppe.

- Créer un subpatch vide nommé "gene-env" en insérant un objet `patcher gene-env`. Tout objet commençant par "patcher" crée un subpatch vide.
- Le remplir avec notre générateur d'enveloppe précédemment créé (copier/coller fonctionne très bien).
- Tout subpatch communique avec son patch parent via les objets `inlet` et `outlet`.

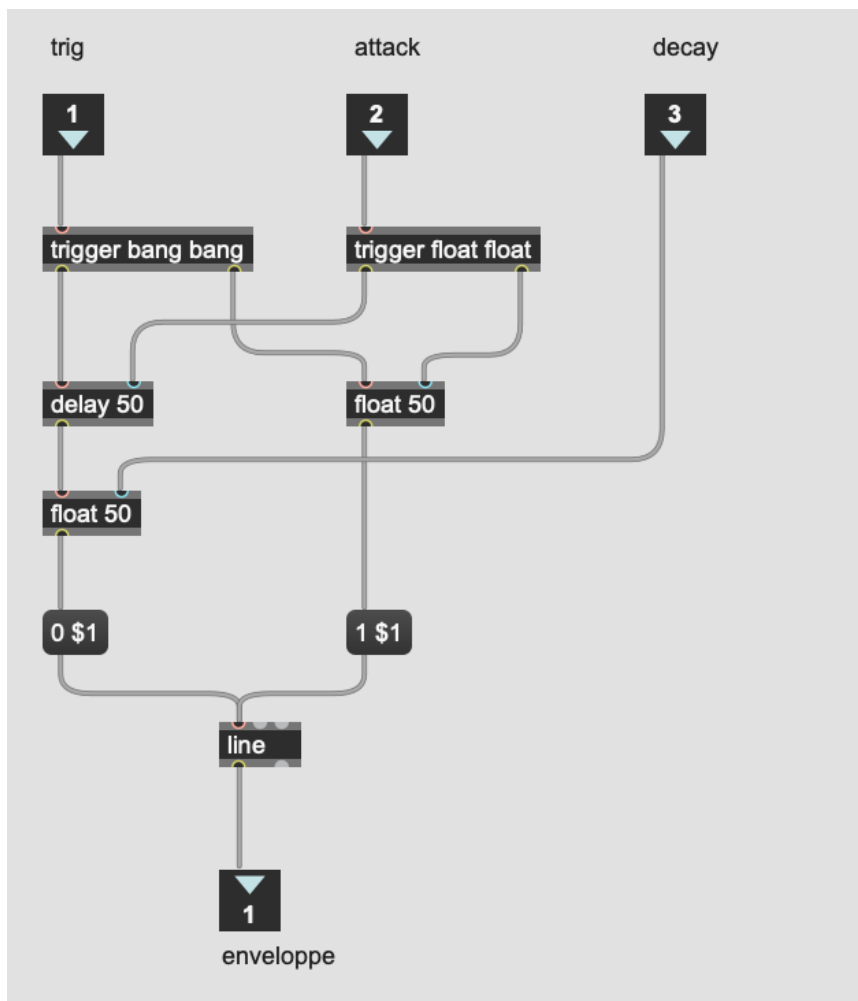


FIGURE 8 – Subpatch du générateur d’enveloppe.

- Connecter `pd gene-env` à votre patch actuel.
- Noter que quelques modifications ont été apportées au générateur d’enveloppe : les objets `float` ont désormais une valeur par défaut de 50ms (utile à l’initialisation du patch lors du chargement). Expliquer à quoi sert `trigger float float` (peut être abrégé en `"t f f"`).

4.7 Subpatch de l’oscillateur (figure 9)

On va faire de même pour l’oscillateur et créer un subpatch via l’objet `patcher oscillator`. Il possède deux inlet : l’un fixe le pitch (la hauteur en Hz), le second l’épaisseur du son (fatness) via le désaccord des oscillateurs.

- Cabler le subpatch et le relier au reste de votre patch ; tester les fréquences de chaque oscillateur en utilisant des objets `print`

- Il vous faudra ajouter deux nouveaux objets "number (f)" : un pour le pitch, l'autre pour le réglage de fatness ;
- En quoi le bloc `trigger bang bang float float` (abrégé en "t b b f f") permet-il de mettre à jour les paramètres "pitch" et "fatness" en temps réel lorsque les sliders sont modifiés ?

4.8 Control d'amplitude

On va utiliser le générateur d'enveloppe créé précédemment pour contrôler désormais l'amplitude.

- Copier/coller le subpatch du générateur d'amplitude ;
- Le connecter, d'une part à deux nouveaux sliders, "Amp-attack" et "Amp-decay", d'autre part à l'inlet 2 de l'objet `*~`.
- Vous réutiliserez la sortie du bouton déclencheur du générateur d'enveloppe de filtre.

4.9 Réception de messages MIDI

Pour piloter directement le patch via un clavier MIDI, il suffit d'insérer, avant le contrôle de pitch, les objets suivants :

- `notein` lit des messages MIDI depuis l'entrée MIDI standard (cf réglage MIDI dans le menu Media) et filtre les notes ; le numéro de note est disponible sur l'outlet 1 ;
- `mtof` convertit un numéro de note MIDI en fréquence.



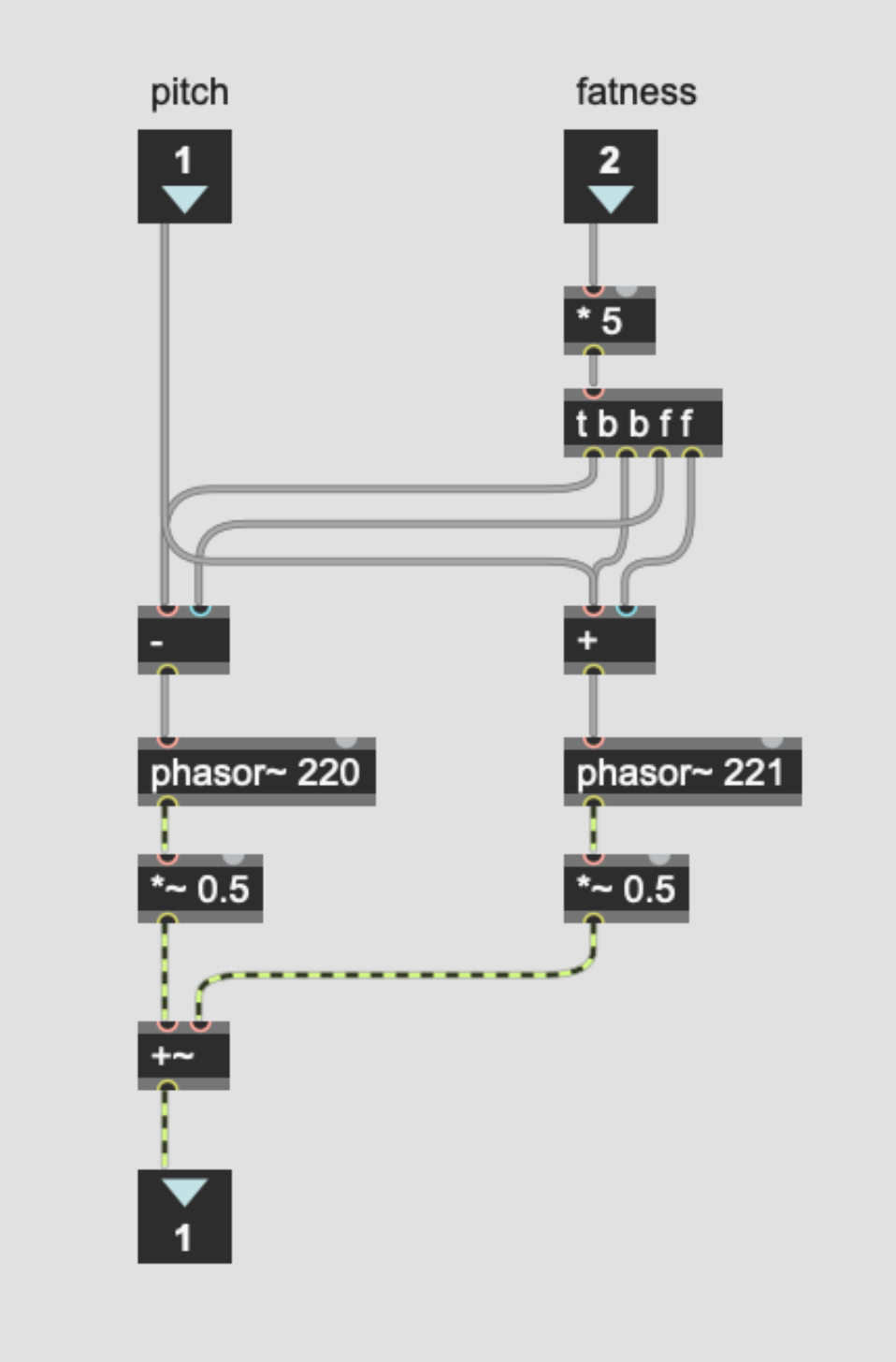


FIGURE 9 – Subpatch de l'oscillateur.