Introduction à la Synthèse sonore

Cours 3 Modèles physiques et granulation

Matthias Puech
matthias.puech@lecnam.net

STMN — Cnam ENJMIN, Angoulême

21-22 novembre 2017

Synthèse sonore

Modèles physiques

- Synthèse de la voix
- Synthèse modale
- Guides d'onde

Sampling et granulation

- Sampling
- Synthèse granulaire

Synthèse par modèle physique

La puissance des machines permet de simuler les processus physiques qui créent les vibrations sonores

$$\text{modèles physiques} \longrightarrow \text{simulation} \longrightarrow \text{algorithmes}$$

Synthèse par modèle physique

La puissance des machines permet de simuler les processus physiques qui créent les vibrations sonores

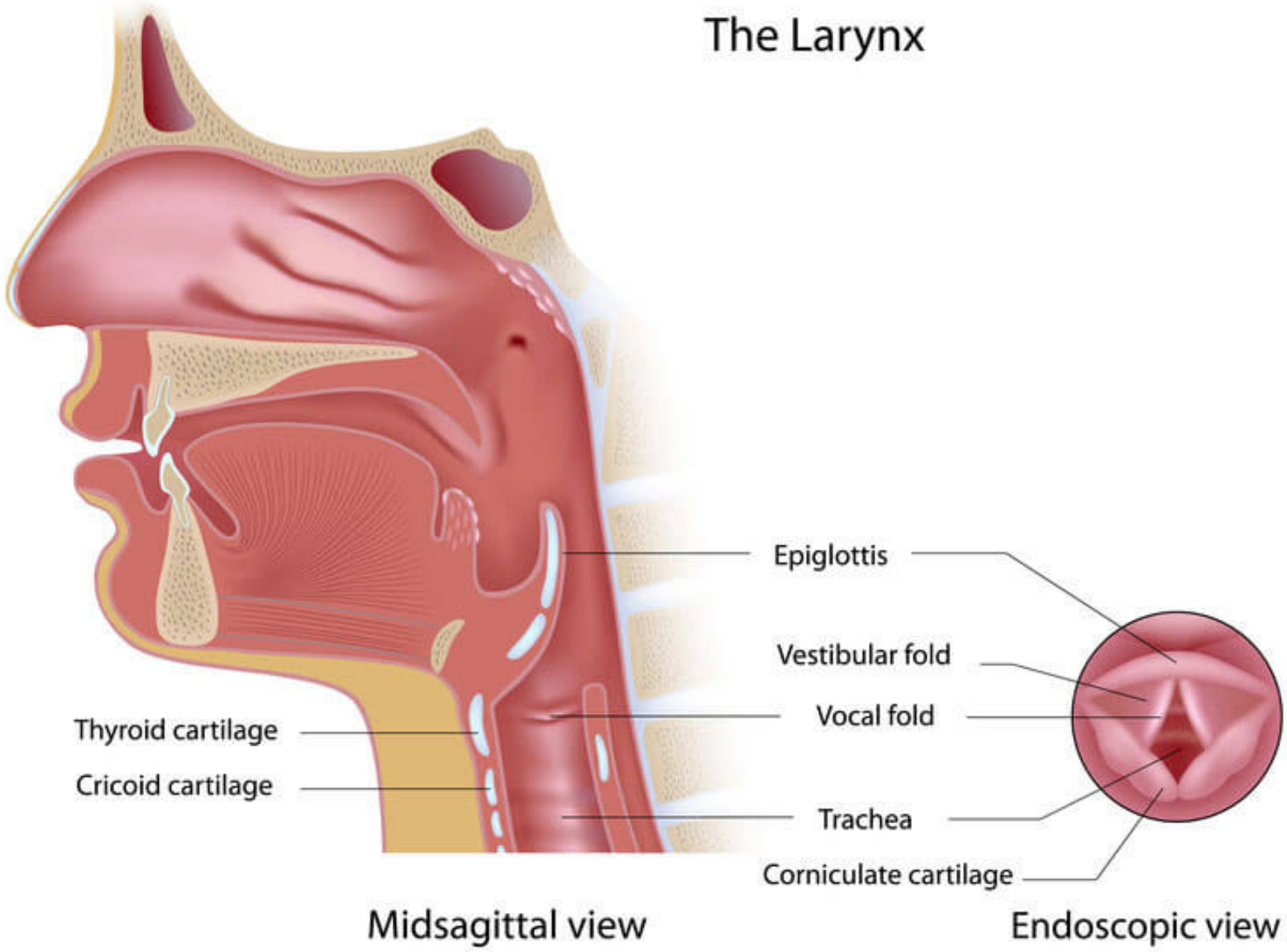
$$\text{modèles physiques} \longrightarrow \text{simulation} \longrightarrow \text{algorithmes}$$

Les familles d'algorithmes classiques

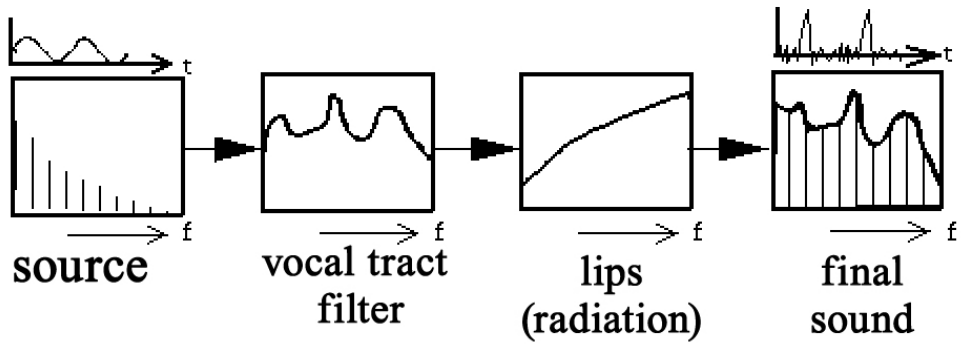
- modèles source-filtre (voix humaine)
- synthèses modale (percussions notamment)
- guides d'onde (instruments à corde)

Synthèse de la voix

Anatomie de l'appareil vocal



Modèle source-filtre



source dent de scie / train d'impulsion
($\sim 50 - 200$Hz)

canal vocal filtre linéaire
(dépendant de la position du larynx)

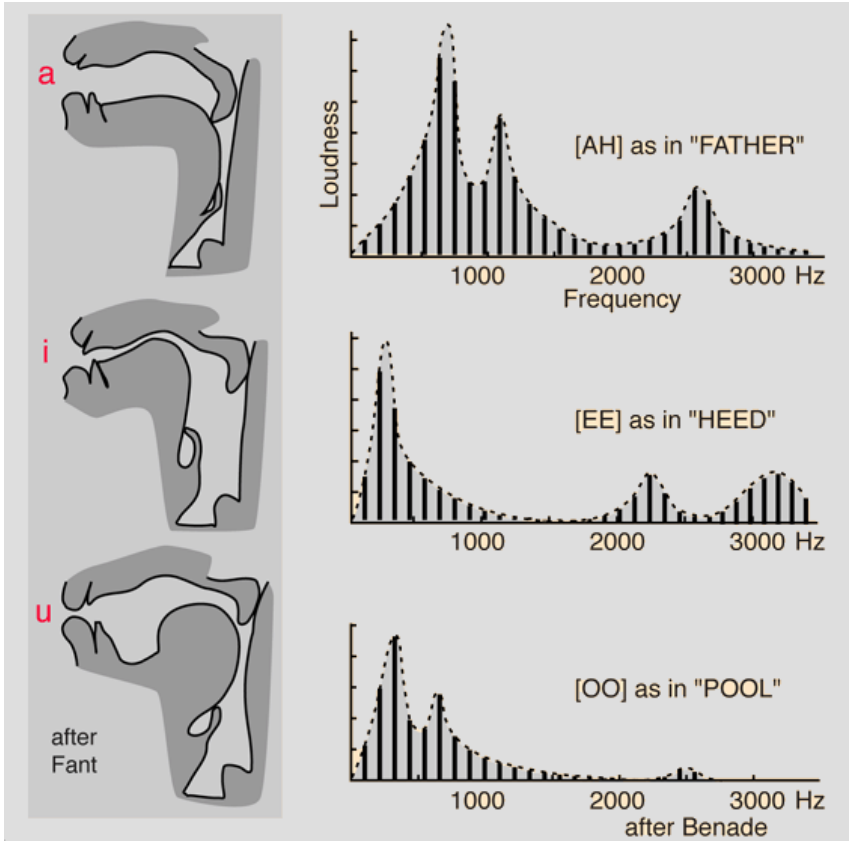
lèvres et dents source de bruit
(pour les fricatives)

Enveloppe spectrale et formants

Quel est l'enveloppe spectrale du canal vocal ?

Enveloppe spectrale et formants

Quel est l'enveloppe spectrale du canal vocal ?



Formants

Par mesure, on distingue 3 bosses, des *formants*,
dont la fréquence dépend de la position de l'appareil
(une ou des harmoniques amplifiées par résonance)

Algorithmes de synthèse vocale

Une longue histoire des algorithmes de Text-to-Speech (TTS) :

⋮

synthèse formantiques VOSIM (1978), FOF (1989)…
(`https://dood.al/pinktrombone/`,
`https://www.youtube.com/watch?v=7LGnozlwU1o`)

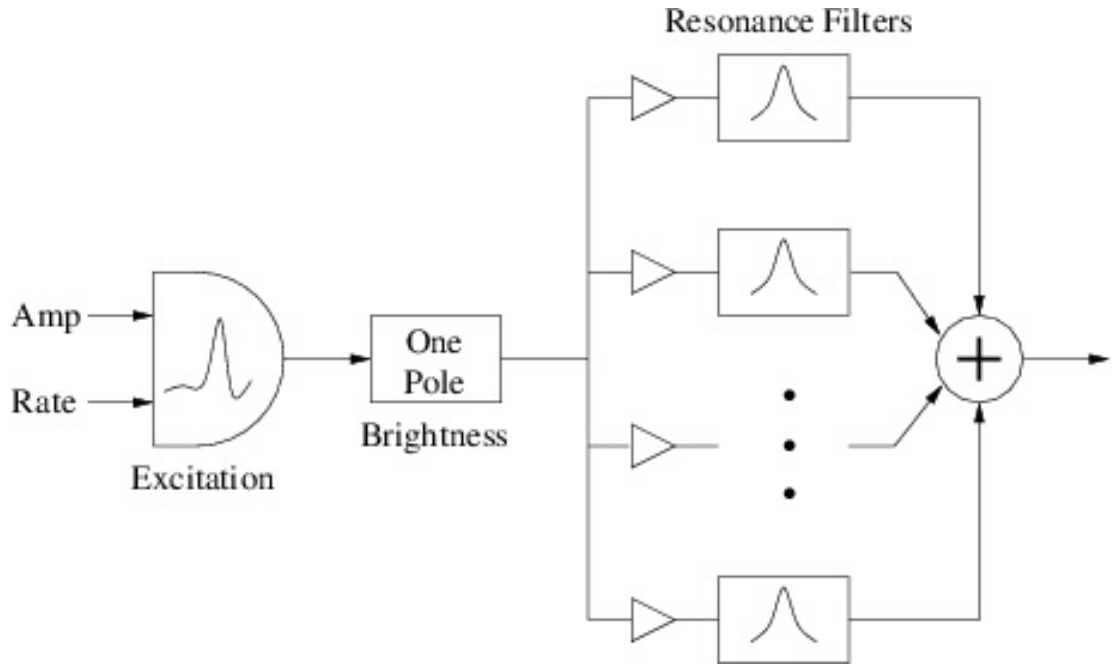
synthèse concaténative concaténation de courts enregistrements
(le plus courant ; proche des techniques granulaire)

syntèse par réseaux de neurones entraîner un RNN sur de la voix ;
lui faire prédire de l'audio sample-à-sample
(`https://deepmind.com/blog/`
`wavenet-generative-model-raw-audio/`)

⋮

Synthèse modale

Simulation de la propagation d'ondes dans des matériaux :



- excitation par impulsions bruitées
  (archet, maillet etc.)
- résonance par banque de filtres passe-bande
  (accordés selon le matériau)

Quelques logiciels

Modalys (Ircam)

`http://instrum.ircam.fr/modalys/`

Chromaphone (AAS)

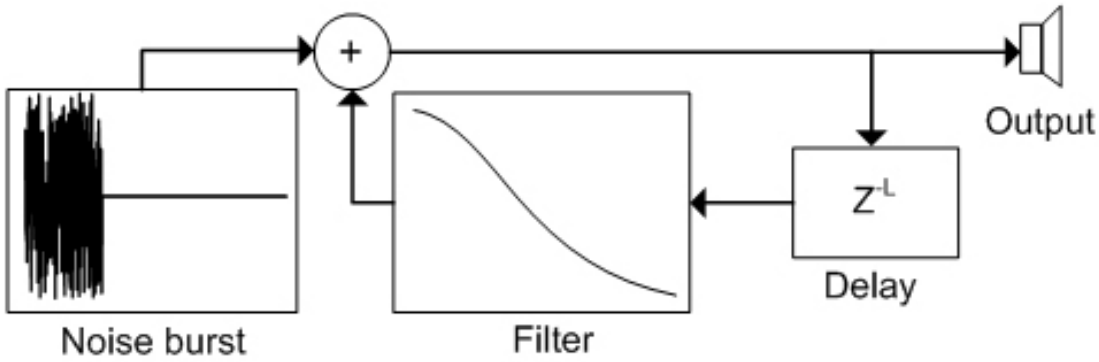
`http://applied-acoustics.com/chromaphone-2/`

Exemple

`https://youtu.be/RdoGlKOvzKk?t=6m35s`

Corde pincée : l'algorithme de Karplus-Strong (1983)

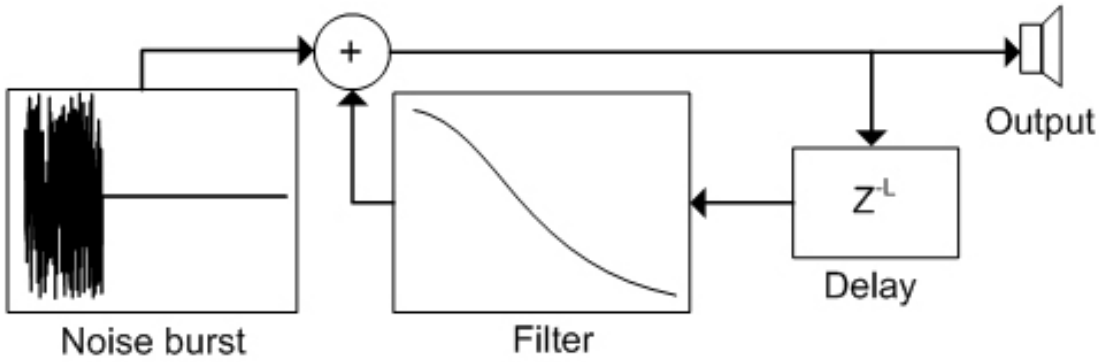
Un algorithme simple et populaire de synthèse de cordes pincées :



- une impulsion courte de buit blanc (∼10ms)
- dans un délai avec réinjection (temps de délai $= \frac{1}{f}$)
- un filtre passe-bas dans la boucle de réinjection

Corde pincée : l'algorithme de Karplus-Strong (1983)

Un algorithme simple et populaire de synthèse de cordes pincées :



- une impulsion courte de buit blanc (∼10ms)
- dans un délai avec réinjection (temps de délai $= \frac{1}{f}$)
- un filtre passe-bas dans la boucle de réinjection

C'est la partie émergée de l'iceberg

Pour aller plus loin : les *guides d'onde*
(Julius O. Smith, `https://ccrma.stanford.edu/~jos/swgt/`)

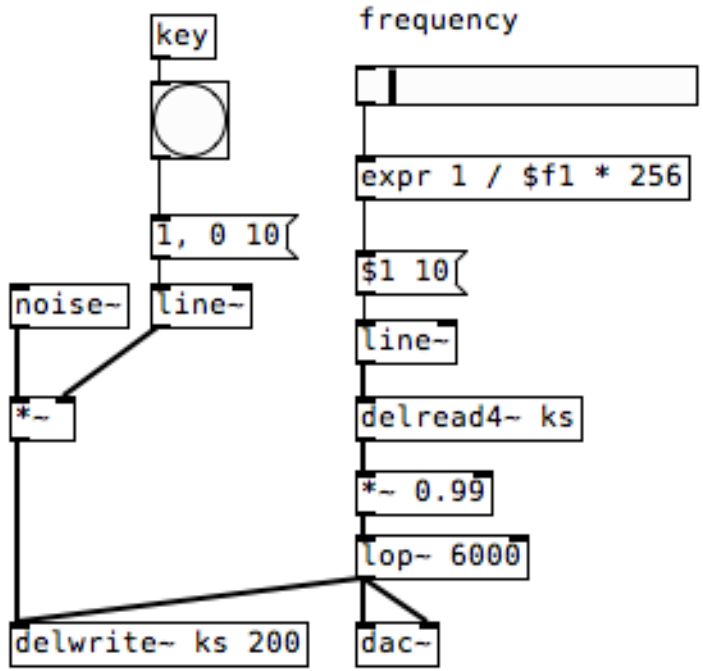
Corde pincée : l'algorithme de Karplus-Strong (1983)

Délais en Pure Data

`delwrite~` écrit son signal d'entrée dans un tableau, en continu

`delread4~` lecture dans tableau n ms après la tête d'écriture

`[20.karplus.pd]`



Synthèse sonore

Échantillonnage (*sampling*)

Dans son sens musical

- enregistrement de fragments audio (“samples”)
- lecture en temps réel à vitesse variable
- modulation et traitement divers
  (filtrage, amplitude, …)

Échantillonnage (*sampling*)

Dans son sens musical

- enregistrement de fragments audio (“samples”)
- lecture en temps réel à vitesse variable
- modulation et traitement divers
  (filtrage, amplitude, …)

Un résolution force brute du problème de la synthèse

polyphonie, multi-échantillonnage massif : des orchestres entiers !
(`https://www.youtube.com/watch?v=FIKlrcgJnvA`)

Pure Data : lire des échantillons

- On stocke les échantillons dans les *tableaux*
  (que nous connaissons déjà)
- On les lit avec `tabread~`
  (que nous connaissons déjà)

`soundfiler` lit un fichier `.wav` ou `.aiff` sur le disque et le stock dans un tableau

argument	nom du tableau
entrée	message “`read -resize fichier.wav`”
sortie	nombre de samples lus

Pure Data : lire des échantillons

Nouveauté

- une boîte *message* dans laquelle on met plusieurs messages séparés par des “`,`” émettra ses messages, dans l’ordre
- donc si j’envoie “`0, 44100 1000`” à un `line~`, il va émettre un segment de signal linéaire allant de 0 à 44100 en 1 seconde (une *rampe*)

Pure Data : lire des échantillons

Nouveauté

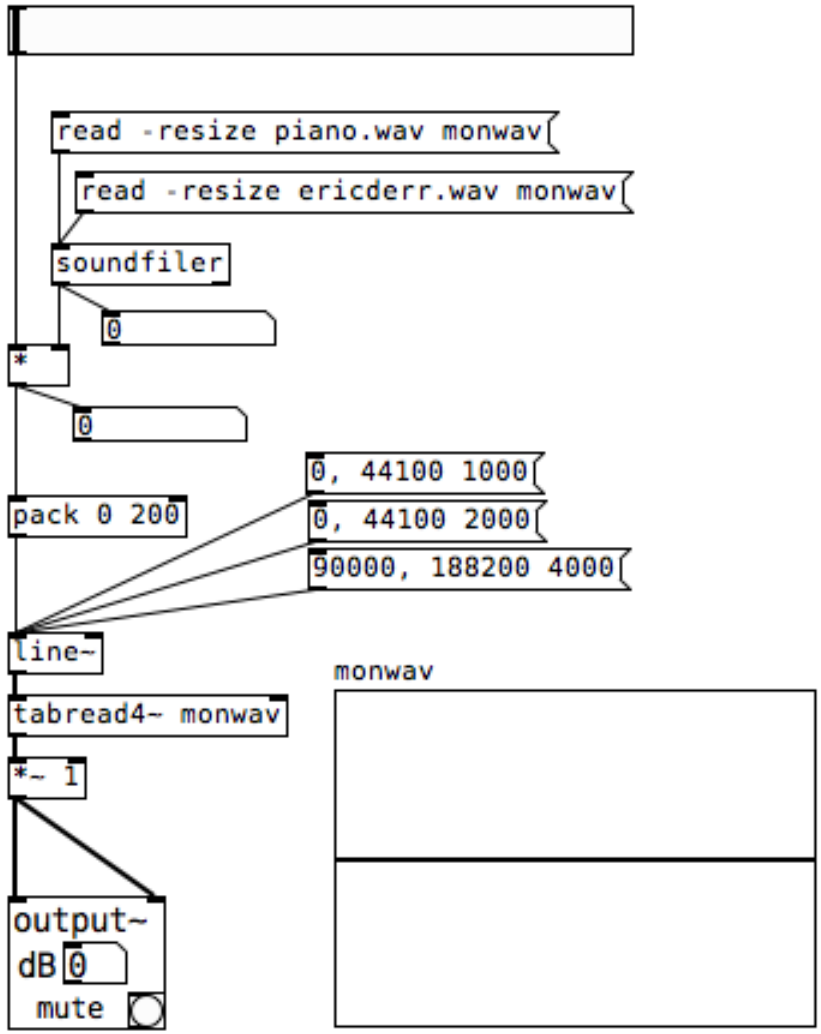
- une boîte *message* dans laquelle on met plusieurs messages séparés par des “`,`” émettra ses messages, dans l’ordre
- donc si j’envoie “`0, 44100 1000`” à un `line~`, il va émettre un segment de signal linéaire allant de 0 à 44100 en 1 seconde (une *rampe*)

Pourra vous être utile

`samplerate~` émet la fréquence d’échantillonage courante quand `bang`é

Pure Data : lire des échantillons

[21.wavread.pd]



Un sampler basique [22.sampler.pd]

On peut donc lire des segments audio à vitesse variable

Un sampler basique [22.sampler.pd]

On peut donc lire des segments audio à vitesse variable

Nouveauté

- dans une boîte *message*, $*i* est remplacé par le *i*-ème élément de la liste entrante

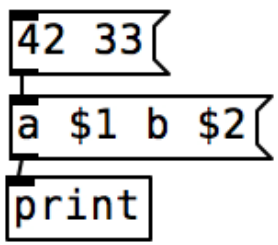
Un sampler basique [22.sampler.pd]

On peut donc lire des segments audio à vitesse variable

Nouveauté

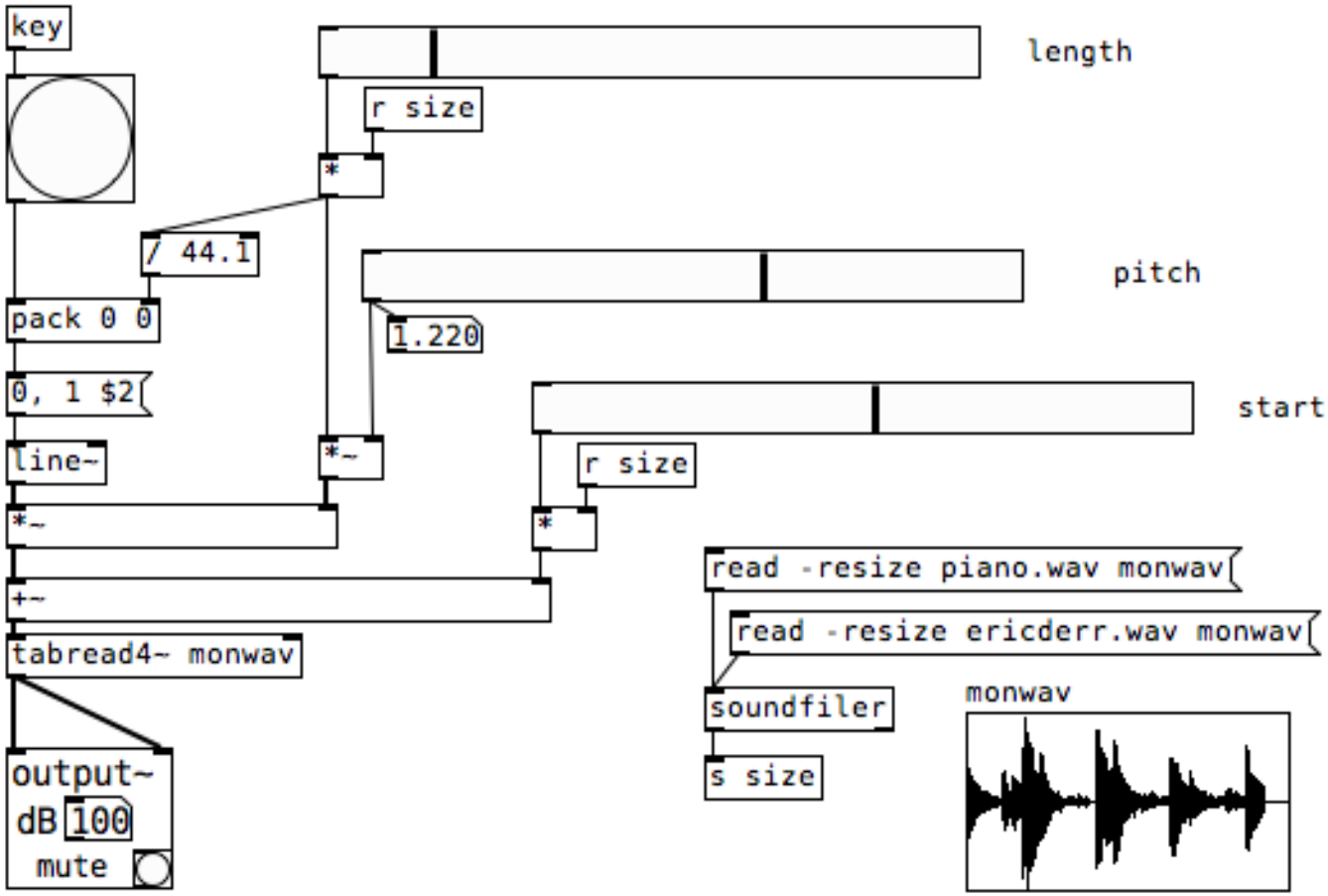
- dans une boîte *message*, $\$i$ est remplacé par le i-ème élément de la liste entrante

Exemple



affiche la liste “`a 42 b 33`”

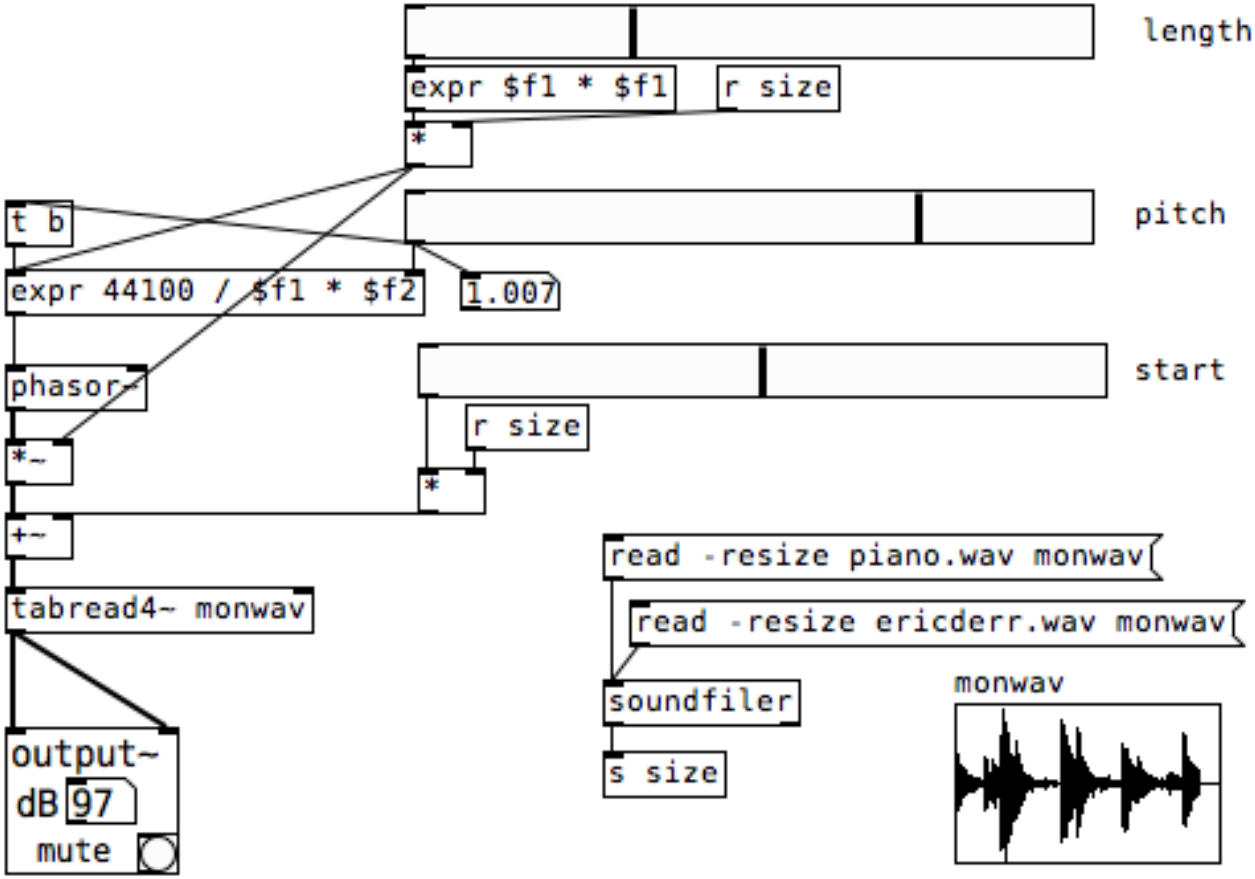
Un sampler basique [22.sampler.pd]



On génère avec `line~`, `+~` et `*~` un segment qui va de `start` à `start` + `length` × `pitch` en `length` × f_S secondes.

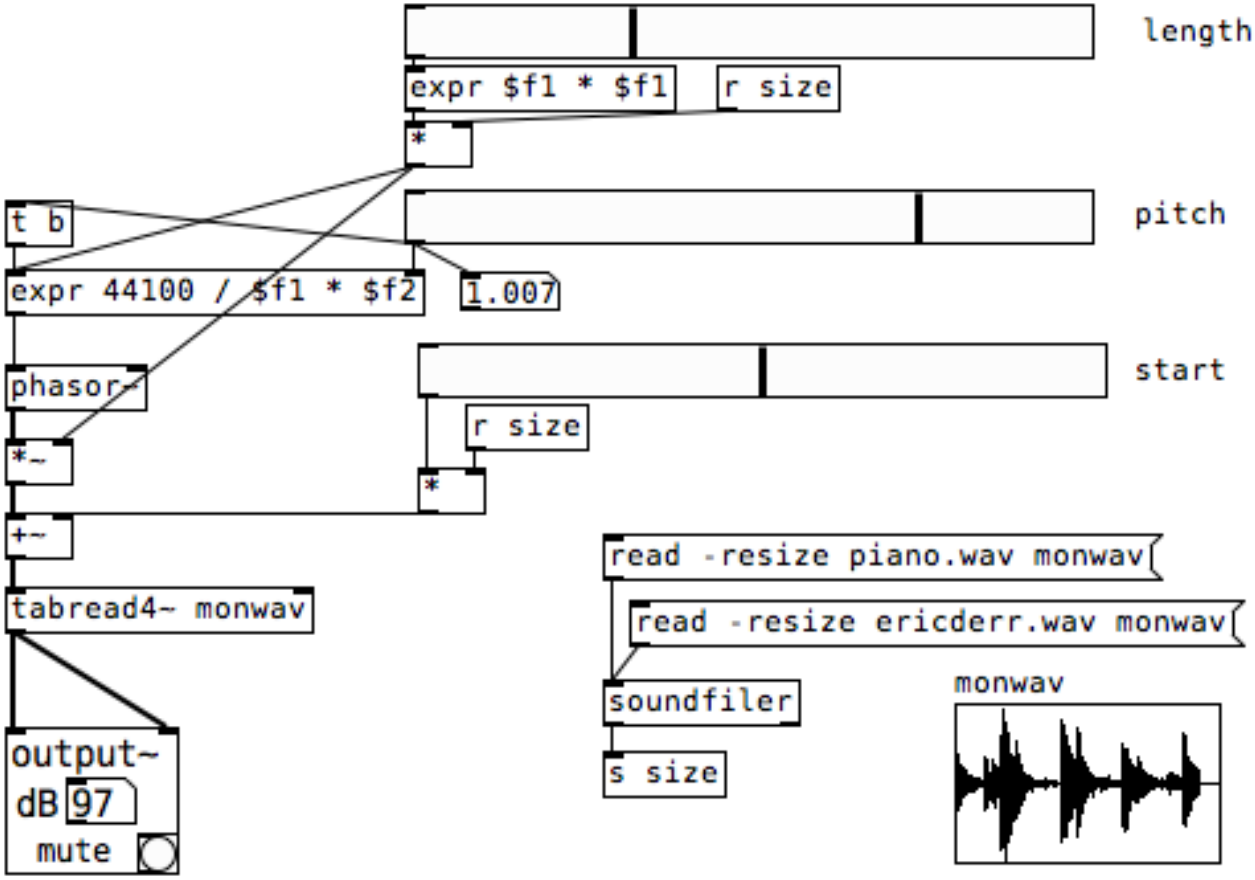
Variante : le looper [23.looper.pd]

On génère le signal linéaire avec `phasor~` ⇝ lecture en boucle



Variante : le looper [23.looper.pd]

On génère le signal linéaire avec `phasor~` ⇝ lecture en boucle



Remarque

On a des clics audibles aux limites du sample

⇝ *fenêtrage* (enveloppe temporelle)

Synthèse granulaire

Concept de Curtis Roads (198 ?)

- lecture de multiples *grains* = échantillons courts (1–100ms)
- les grains peuvent se superposer (polyphonie nécessaire)
- pour chaque grain, paramètres :
  - forme d'onde (échantillon),
  - longueur,
  - enveloppe d'amplitude (*fenêtrage*),
  - vitesse de lecture (pitch)
  - filtrage, ...

Synthèse granulaire

Concept de Curtis Roads (198 ?)

- lecture de multiples *grains* = échantillons courts (1–100ms)
- les grains peuvent se superposer (polyphonie nécessaire)
- pour chaque grain, paramètres :
  - forme d'onde (échantillon),
  - longueur,
  - enveloppe d'amplitude (*fenêtrage*),
  - vitesse de lecture (pitch)
  - filtrage, ...

Exemple

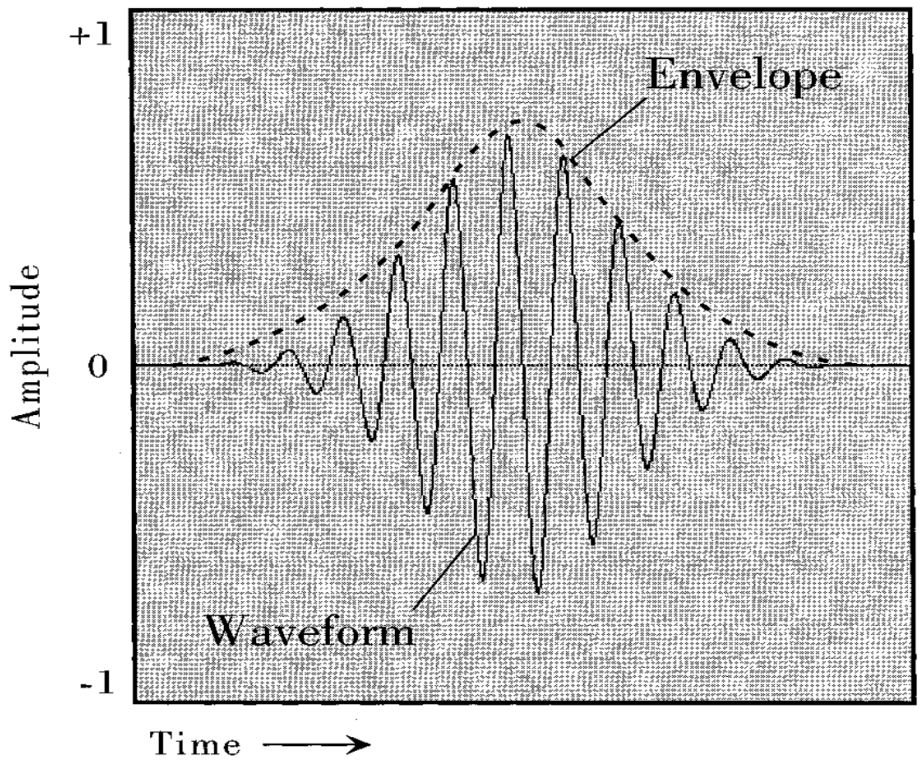
C. Roads, Half Life (1999)
(`https://www.youtube.com/watch?v=70byQuA58fg`)

Synthèse granulaire

Curtis Roads — Microsound (2001)

Grains et fenêtrage

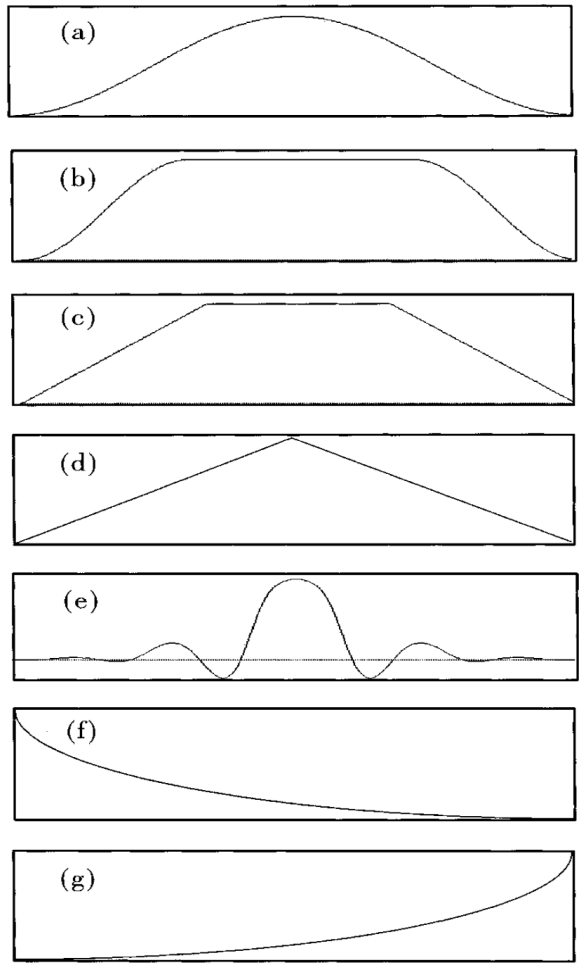
Un grain



Grains et fenêtrage

Fenêtrage

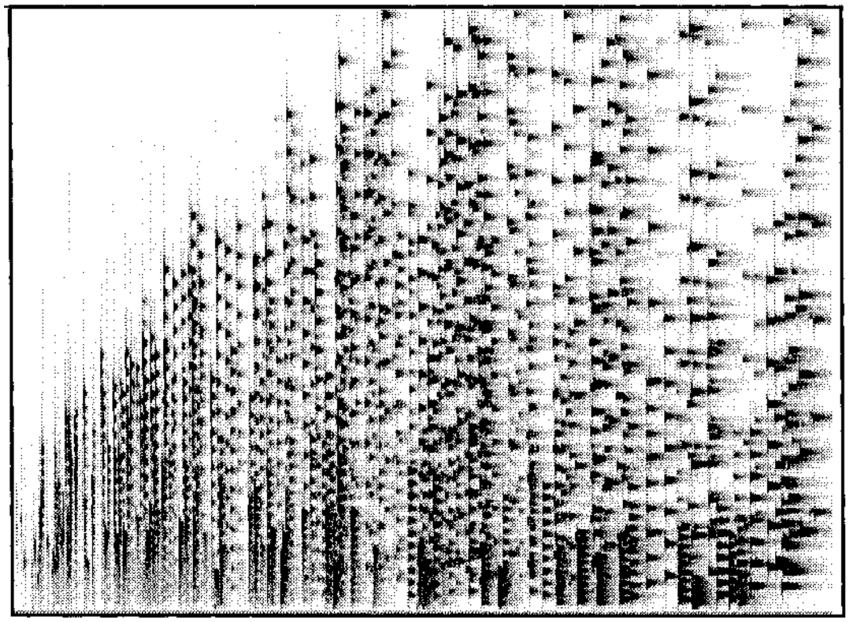
= enveloppe temporelle $(0 \rightarrow 1 \rightarrow 0)$



Applications

Nuages de grains

génération de textures sonores inédites

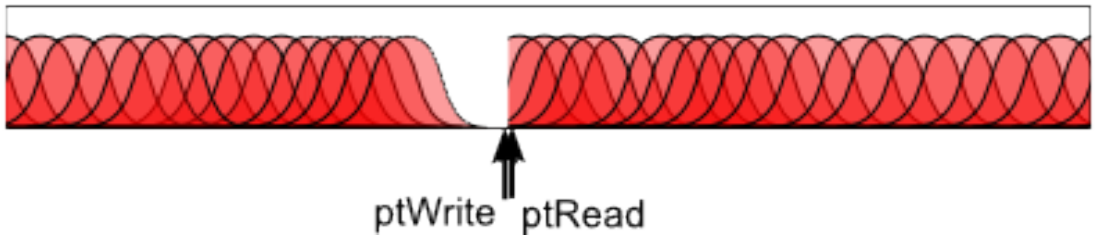


des grains identiques sont aléatoirement distribués dans le temps ;
on contrôle les paramètres de cet aléa (distribution, centroïde...)

Applications

Pitch shifting / time stretching

a.k.a. Overlap/Add



on superpose à intervalle régulier des grains d'un même son
on contrôle la vitesse de lecture du chaque grain (pitch)
et la vitesse de lecture globale (temps)

Exemples

- moteur de voiture dans un jeu vidéo
- effets sur la voix

Synthétiseur granulaire [25.granular.pd]

Nouveautés

`spigot` laisse passer un message si son état est 1, le bloque sinon

- entrée 1 : le message à passer (ou pas)
- entrée 2 : fixe l'état (1 ou 0)

`loadbang` envoit `bang` au chargement du patch

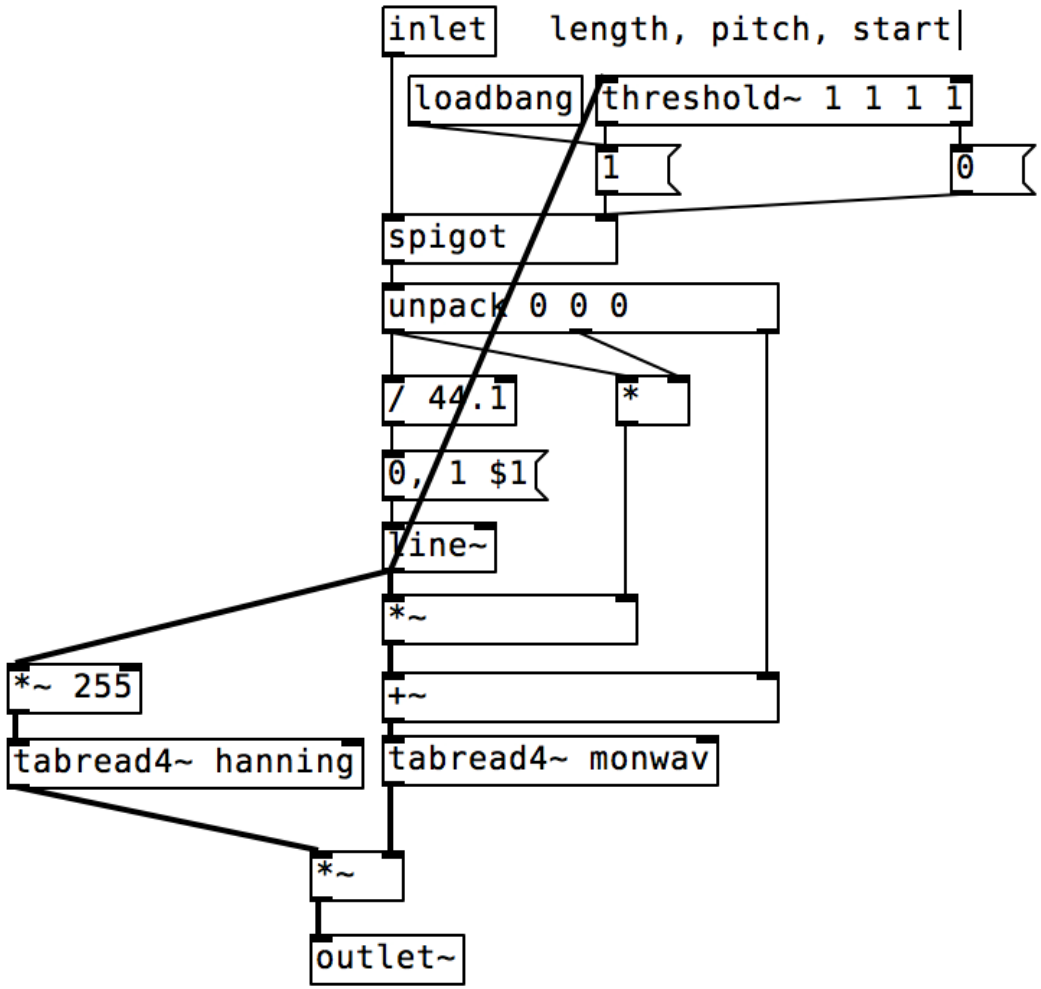
`unpack` le contraire de `pack` :

- reçoit une liste de *n* éléments
- sort indépendemment les *n* élément
- arguments : le nombre de sorties et leurs valeurs par défaut

`threshold` envoit *bang* quand le signal d'entrée passe un seuil

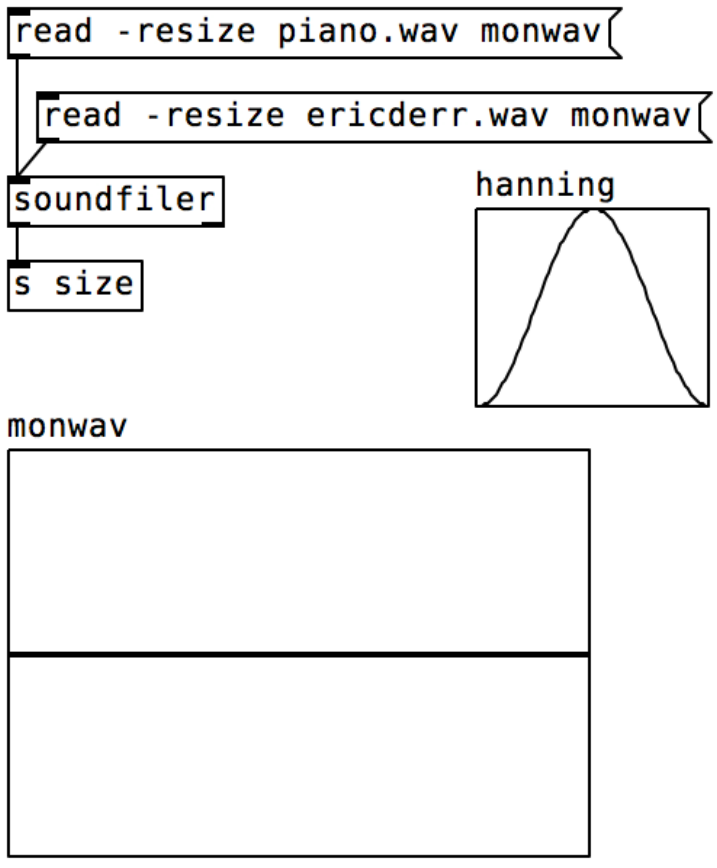
Synthétiseur granulaire [25.granular.pd]

Le lecteur de grain [gran-voice~.pd]



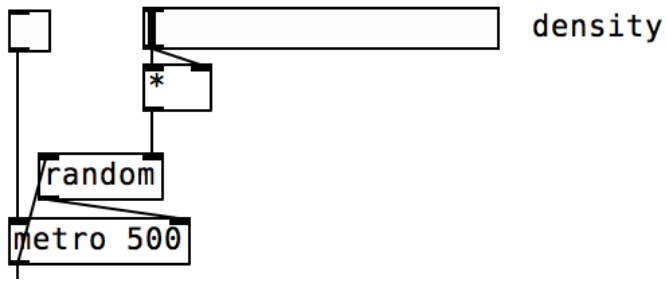
Synthétiseur granulaire [25.granular.pd]

Les tables



Synthétiseur granulaire [25.granular.pd]

Génération de bangs aléatoires



Synthétiseur granulaire [25.granular.pd]

Le tout