

Mettez vos casques et attention au volume sonore...

Sur mon Mac, le répertoire rsound est ~/Library/Racket/6.6/pkg/rsound/rsound/

*Ne touchez pas à ces fichiers, faites-en éventuellement une **COPIE** dans votre répertoire de travail.*

1. Le format WAVE PCM 16 bits

Pour votre première rencontre avec le traitement du son en Racket avec **rsound**, nous commençons par nous intéresser à la fréquence d'échantillonnage d'un son. Normalement elle est de 44.1Kz, soit 44100 échantillons par seconde. Bien entendu, dans l'industrie musicale, elle pourra être meilleure : 48Kz, 88.2Kz jusqu'à 192Kz ! Il semble que les studios professionnels sont satisfaits avec 48Kz/24 bits. **Rsound** utilise pour l'instant le format **WAVE PCM 16 bits à 44.1Kz**. Il peut lire des fichiers audio échantillonnés par exemple à 8000Hz lors de l'enregistrement de voix humaine, mais il reste strict sur le format PCM (le 20^{ème} octet du fichier¹ indique s'il s'agit bien de PCM).

Exercice 3.1 Avec le navigateur conseillé *Chrome*, allons sur le Web rechercher un fichier .wav échantillonné à 8000 Hz. Il contiendra des nombres lus, par exemple le fichier 27.wav contiendra : *twenty seven*. L'enregistrement n'est pas très bon, peu importe. Allez le trouver sur Voxeo :

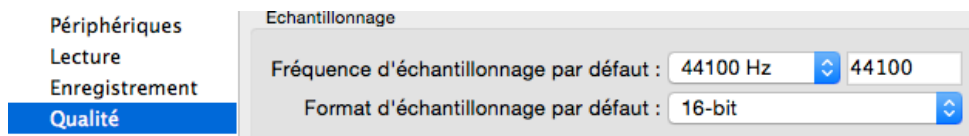
<https://evolution.voxeo.com/library/audio/prompts/numbers/index.jsp>

et récupérez le fichier 27.wav dans votre répertoire de travail. Ecoutez-le.

Exercice 3.2 Maintenant vous êtes dans Racket, ouvrez le fichier TP3-1.rkt et définissez une variable n27 dont la valeur est le son obtenu en chargeant 27.wav dans la mémoire Scheme. Comme vous pouvez le constater, ceci aboutit à une erreur dont le message est clair : ce n'était pas un fichier PCM². On ne peut rien faire du côté de Racket, nous utiliserons un programme externe **Audacity** déjà installé :

<http://audacity.sourceforge.net/?lang=fr>

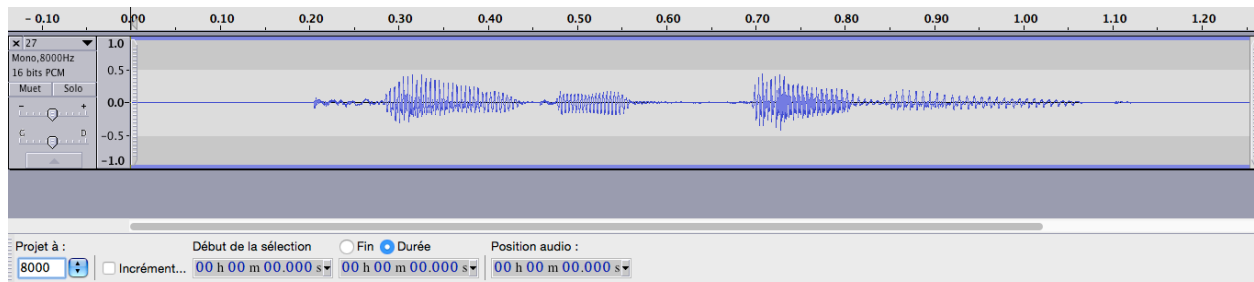
Dans les Préférences d'Audacity, réglez la qualité par défaut :



Avec Audacity, ouvrez le fichier 27.wav. Les 8000 Hz (coin en bas à gauche) ne sont pas un problème pour Racket qui saura les modifier (on pourrait le faire dans Audacity, mais ne le faites pas maintenant).

¹ <http://soundfile.sapp.org/doc/WaveFormat/>

² Il y a des sites qui fournissent du format PCM 16 bit (mais peut-être à 8000 Hz) : <https://www.freesound.org>



Demandez *Exporter...* depuis le menu *Fichier* et sauvez dans le répertoire courant sous le format **WAV (Microsoft) signé 16 bits PCM**, et zou ! Nommez le nouveau fichier **27-pcm.wav**.

Exercice 3.3 A partir de maintenant nous travaillons dans Racket. Vous devriez pouvoir charger le nouveau fichier **27-pcm.wav** sans erreur dans la variable **n27** de type **rsound**. Ecoutez-le avec **play**.

- Vérifiez que le fichier **27-pcm.wav** est bien échantillonné à 8Kz.
- Comment pouvez-vous le vérifier directement en mémoire sur la variable **n27** ?
- Combien comporte-t-il de *frames* (échantillons), avec **rs-frames** ou directement.
- Programmez une fonction (**duration snd**) retournant la durée inexacte du son en secondes. Calculez la durée du son **n27** [rép: 1.2545 sec]

- Exercice 3.4**
- Avec **rs-draw**, faites afficher la courbe du son. Visualisez-vous les quatre syllabes ?
 - Avec **clip**, éliminez les segments horizontaux au début et à la fin qui sont inaudibles (trouvez les frontières à la souris). Nommez le nouveau son **n27-clipped** et enregistrez-le sur le disque sous le nom **27-pcm-clipped.wav**. Faites afficher la nouvelle courbe. Quelle est la durée du nouveau son ?
 - Faites jouer le nouveau son. Si vous n'avez pas commenté le **(play n27)** antérieur, il risque de se superposer au **play** actuel car **play** est asynchrone. Dans ce cas, il faut faire suivre le premier d'un **(sleep n)** où **n** est au moins la durée du son joué. Oui, *c'est un peu désagréable, mais nous aurons mieux avec les pstreams...*

Exercice 3.5 En travaillant uniquement avec la variable **n27-clipped**, définissez une autre variable **n27b7** dont la valeur est le son *twenty seven seven*.

Exercice 3.6 Utilisez la fonction **(resample-to-rate new-frame-rate sound)** et regardez sa doc en ligne. Comment l'utiliseriez-vous pour ré-échantillonner le son **n27b7** pour produire un nouveau son **n27b7-44100** puis un fichier **n27b7-44100.wav** ? Vérifiez avec Audacity que l'échantillonnage est bien 44.1Kz et surtout que le son est bien le même à l'oreille ! Comparez aussi les courbes des deux sons. Avec Audacity, transformez le fichier wav en fichier mp3.

2. Utilisation des pstreams.

Exercice 3.7 Les 7 sons percussifs suivants sont définis par **RSound** :

- bassdrum** : grosse caisse
- snare** : caisse claire
- crash-cymbal** : cymbale crash à son explosif
- c-hi-hat-1, c-hi-hat-2** : cymbale à pied ou Charleston, formée de deux cymbales en contact

clap-1, clap-2 : claquement de mains

Rien ne vous empêche bien entendu de télécharger d'autres sons percussifs (courts) au bon format WAV-PCM16, ou de les définir vous-mêmes en enregistrant vos bruits de casseroles...

- a) Avec une boucle `for`, `play` et `sleep`, faites jouer chacun de ces sons, séparés par 0.1 seconde.
- b) Définissez la variable `percussions` dont la valeur est le son obtenu en concaténant tous ces 7 sons, chacun étant séparé du suivant d'un silence de 0.1 seconde. Ecoutez-la ensuite avec `play`.
- c) A partir de maintenant nous n'utilisons plus `play`, mais une `pstream ps`, définissez-la dans l'éditeur.
- d) A l'aide de `pstream-queue`, programmez une fonction (`stream-percussions`) qui envoie dans la `pstream ps` les 7 sons percussifs, en séquence et séparés par 0.1 seconde. Parcourez itérativement la liste des 7 sons et envoyez chacun dans la `pstream` en indiquant l'heure à laquelle il sera joué...
- e) Avec `pstream-queue-callback` (lisez sa doc en ligne) faites en sorte que ces 7 sons jouent en boucle dans la `pstream`. Programmez à cet effet la fonction (`loop-percussions`).

3. Un séquenceur pas à pas

Un logiciel professionnel de musique et de traitement du son s'appelle un **DAW** (*Digital Audio Workstation* ou *station audionumérique*). Certains DAW sont gratuits, d'autres très chers. Apple fournit pour les amateurs³ GarageBand (gratuit sur Mac et 5€ sur iPhone/iPad) et l'on peut brancher un clavier ou une guitare électrique. Le DAW fournit nombre d'outils intégrés pour traiter le son, et peut abriter des modules externes, gratuits ou payants : les **VST** (*Virtual Studio Technology*, pour Linux, Mac ou Windows) et les **AU** (*Audio Units*) pour Mac uniquement. Parmi ces outils se trouve le **séquenceur pas à pas**, permettant de construire une séquence audio avec des sons déjà existants (notes d'instruments, bruits, percussions). Avec des sons percussifs, on parle de **boîte à rythmes** (*drum machine*), regardez à la fin de ce document l'écran du Roland TR-707 de 1985.

Exercice 3.8 Le prix du logiciel *Reaper* (<http://www.reaper.fm>) est très bas (\$60), mais vous pouvez le **télécharger** et l'utiliser temporairement en mode démo. Achetez-le si vous souhaitez disposer d'un DAW bon marché. Dans la vidéo que vous allez visualiser, je construis avec *Reaper* et un petit synthétiseur intégré un morceau de musique qui va être joué en boucle. La boucle est automatique et faite par le séquenceur. En musique, une **boucle** (*loop*) est un morceau de musique joué de manière répétitive en arrière-plan. Un DAW contient souvent beaucoup de boucles, et on peut en télécharger de nouvelles (celles de GarageBand sont nombreuses et excellentes). Regardez la vidéo sur :

<http://deptinfo.unice.fr/~roy/PF2/sequencer.mp4>

- Nous ne construirons pas une interface graphique pour le séquenceur, une animation Racket *sans graphisme* (!) suffira, juste pour l'horloge, le canvas 100 x 100 par défaut ne servant à rien.

Exercice 3.9 Voici une première version de l'animation, que vous allez améliorer. Copiez-la dans votre éditeur Racket, comprenez-la et exécutez-la. J'utilise une `pstream` au lieu de `play`, c'est mieux.

```
(require 2htdp/image 2htdp/universe) ; pour images et animations
(define ps (make-pstream))           ; un flot audio
(define (sequencer_v1)
  ; (snare bassdrum snare c-hi-hat-1) à 120 bpm (beats par minute)
  (define BACKGROUND (rectangle 200 200 'solid "yellow"))
```

³ Le DAW professionnel d'Apple sur Mac se nomme LogicPro, voir <https://www.apple.com/fr/logic-pro/>

```

(define SOUNDS (list snare bassdrum snare c-hi-hat-1))
(define clock-freq 1/2) ; fréquence d'horloge : 2 beats/seconde = 120 bpm
(define INIT 0) ; le monde est le nombre de tops d'horloge
(define (suivant t) ; monde -> monde
  (case (modulo t 4)
    ((0) (pstream-play ps snare))
    ((1) (pstream-play ps bassdrum))
    ((2) (pstream-play ps snare))
    (else (pstream-play ps c-hi-hat-1)))
  (+ t 1))
(define (dessiner t) ; suivant est optionnel mais pas dessiner !
  BACKGROUND)
(big-bang INIT
  (on-tick suivant clock-freq (/ 30 clock-freq)) ; pendant 30 secondes...
  (on-draw dessiner)))

```

Exercice 3.10 Vous allez maintenant produire une version plus générale (`sequencer_v2 L bpm`) où `L` est une **liste de sons**, et `bpm` est le nombre de battements par minute (à chaque battement on fait jouer un son de `L`, en bouclant dans `L`). Il faudra vous débarrasser du `case` et le remplacer par un seul appel plus général à `pstream-play`. Le texte sera donc plus court que `sequencer_v1`. Essayez cette nouvelle version avec :

```

(sequencer_v2 (list snare bassdrum snare c-hi-hat-1) 120)
(sequencer_v2 (list snare c-hi-hat-1 snare bassdrum bassdrum) 240)
(sequencer_v2 (list snare c-hi-hat-1 snare bassdrum bassdrum (silence 1)) 240)

```

Exercice 3.11 Jusqu'à présent la scène ne sert à rien. Pourquoi ne pas en profiter pour visualiser un **métronom** ? Dessinez un disque de rayon 50 dont la couleur change à chaque battement : rouge, noir, rouge, noir, etc. En plus, ajoutez un argument `vol` au séquenceur qui permettra de choisir le volume sonore entre 0 et 1 (danger : vérifiez que l'argument est bien dans `[0,1]`). Nommez (`sequencer_v3 L bpm vol`) cette dernière version.



L'ancienne boîte à rythmes Roland-TR707

STEP	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
CYMBAL																
HiHAT			●				●				●				●	
HCP/TAMB																
RIM/COWBELL																
HiTom																
Mid Tom																
Low Tom																
SNARE DRUM					●									●		
BASS DRUM	●				●				●					●		
ACCENT																

Remarques finales : i) Il y avait une autre stratégie pour construire le séquenceur, qui consistait à assembler les sons avec la primitive `assemble` et à faire jouer le son en boucle avec `pstream-queue-callback`. C'était un peu plus technique...

ii) L'idéal bien entendu est de programmer le séquenceur de sorte à laisser l'utilisateur composer sa boucle graphiquement, comme le faisait déjà le Roland ci-dessus en 1985...

iii) Certains font de l'argent en programmant des jeux basés sur les séquenceurs :
<https://www.youtube.com/watch?v=Vs4H3pUBfRE>

