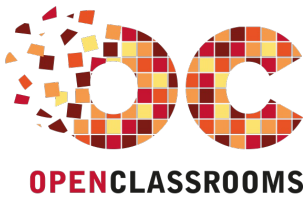
De la musique avec Caml

Par iansus



www.openclassrooms.com

Dernière mise à jour le 12/05/2012

Sommaire

De la musique avec Caml

Par  iansus

Mise à jour : 12/05/2012
Difficulté : Facile Durée d'étude : 1 heure

Salut les zéros ! Aujourd'hui, TP musical !

Ne vous attendez pas à savoir jouer du Beethoven à la fin de ce tutoriel (Beethoven n'avait pas Caml), mais vous pourrez rentrer quelques-unes de vos partitions favorites.

Nous allons pour cela utiliser Caml-Light, dont les liens de téléchargement seront donnés dans la première partie.
Sommaire du tutoriel :

Téléchargement de Caml-Light

Pour la suite de ce cours, vous aurez besoin de Caml, la version Light étant suffisante.
Pour la télécharger, rendez vous sur le site de l'INRIA, et choisissez la version **Caml-Light**.

Ce programme est suffisant, mais peu pratique, c'est pourquoi je vous recommande de télécharger en plus le programme **Wincaml** distribué par *Jean Mouric* sur cette page.

Pour l'installation, rien de plus simple, exécutez le fichier fourni par l'INRIA pour installer Caml-Light sur votre PC, puis dé-zippez Wincaml à l'endroit de votre choix. L'exécutable qui nous servira est à l'intérieur et se nomme *WinCaml.exe*.

Nous voilà fin prêts pour commencer ce cours .

La librairie Graphics, et quelques notes

Comme la plupart des langages, Caml possède une liste fournie de librairies qui peuvent se greffer sur le noyau principal et sur la librairie standard. Celle qui va nous intéresser est la librairie **graphics**, livrée avec les deux versions proposées au téléchargement.

Nous allons donc commencer par charger la librairie **Graphics** en mémoire, avec la commande *open* :

Code : OCaml - Chargement de Graphics

```
(* Début du code *)

#open "graphics";; (* la commande pour charger graphics *)

open_graph "";;   (* Une petite vérification ... *)
close_graph();;   (* pour savoir si la librairie est chargée *)
```

Ayant l'habitude d'utiliser WinCaml qui ne propose pas le "#" avant la commande en entrée, je ne mets jamais de "#" en début de ligne.
Celui qui est donc dans le code devant *open "graphics"* n'est donc pas accidentel et fait partie de la commande.

Intéressons nous à la commande qui va nous permettre de jouer des notes : **sound**
Cette fonction a été *curryfiée*, ce qui signifie que les arguments s'écrivent séparés par un espace, et non entrés comme un couple. Voici sa syntaxe :

Code : OCaml

```
sound fréquence_en_Hz durée_en_ms;;
```

Nous pouvons donc faire quelques essais :

Code : OCaml - Chargement de Graphics

```
(* Début du code *)

#open "graphics";; (* la commande pour charger graphics *)

(* Plus de vérif, si vous êtes ici, c'est que ça marche
correctement *)

sound 440 1000;; (* Un joli LA d'une seconde *)
sound 415 500;;  (* Puis un SOL d'une demi-seconde *)
sound 554 2000;; (* et enfin, un do# de 2 secondes *)
```

Si vous entendez correctement les notes jouées par Caml, c'est que c'est bon, et que vous être prêts à passer à la création de "partitions" !

Première partition

Dans l'exemple qui précédait, je vous ai affirmé que le LA avait une fréquence de 440 Hz, le sol 415 et le Do# 554.
Qu'en sais-je réellement ? Il se trouve que la fréquence de toutes les notes a été enregistrée et reportée sur de nombreux sites, dont celui-là. Vous pourriez vous amuser à toutes les mettre à chaque fois les unes à la suite des autres avec de multiples appels à **sound**, mais votre partition deviendrait illisible très rapidement.

Pour vous épargner cela, nous allons coder un programme qui va prendre en entrée une partition dans un format plus lisible, et donner en sortie la musique correspondante.

Étape 1 : Rentrer les notes

Dans le but de vous enlever le fardeau de rentrer les notes dans des variables séparées, je l'ai fait moi-même, et vous le propose ici :

Code : OCaml - Notes de musique [4 octaves]

```
let do1 = 65 and do2=131 and do3 = 262 and do4=523 and  dod1 = 69
and dod2 = 139 and dod3 = 277 and dod4 = 554
and  re1 = 73 and re2 = 147 and re3 = 294 and re4 = 587
and  mib1 = 78 and mib2 = 156 and mib3 = 311 and mib4 = 622
and  mi1 = 82 and mi2 = 165 and mi3 = 330 and mi4 = 659
and  fa1 = 87 and fa2 = 175 and fa3 = 349 and fa4 = 698
and  fad1 = 92 and fad2 = 185 and fad3 = 370 and fad4 = 740
and  sol1 = 98 and sol2 = 196 and sol3 = 392 and sol4 = 784
and  sold1 = 104 and sold2 = 208 and sold3 = 415 and sold4 = 830
```

```
and  la1=110 and la2 = 220 and la3=440 and la4 = 880
and  sib1 = 117 and sib2 = 233 and sib3 = 466 and sib4 = 932
and  si1 = 123 and si2 = 247 and si3 = 494 and si4 = 988
and  do5 = 1046;;
```

Vous n'aurez qu'à coller cette portion de code au début de votre code, et le tour sera joué

Étape 2 : Définir le format de partition

Le format de partition que je vous propose est adapté au programme qui se construira dans la suite du cours, et est le suivant :

Code : Autre

```
partition : int vect vect
```

En Caml, un **vect** est un tableau, indicé de 0 à n-1, où n est le nombre de cases du tableau. La variable n se récupère à l'aide de la fonction *vect_length*. Le type d'un vect est donc le type de ses éléments, qui doit être unique, suivi du mot clé vect, par exemple, *int vect* ou bien *string vect*.

Pour définir un vect, deux méthodes existent :

Code : OCaml

```
let mon_vect = [|1;4;5;32;0;-7|];; (* On définit toutes les cases du
tableau, ici un int vect *)

let mon_vect = make_vect (n) (a);; (* Crée un tableau de n cases,
toutes initialisée à la valeur "a" *)

mon_vect.(i);; (* accès à la i-ème case du tableau, 0<=i<n *)
```

Le type **int vect vect**, qui doit être lu comme **(int vect) vect** est donc un tableau de tableaux d'entiers, par exemple :

```
let int_vect_vect = [| [|1;2;3|] ; [|4;5;6|] ; [|7;8;9|] |];;
```

En fait, en y regardant bien, ce n'est rien d'autre qu'une matrice, si ce n'est que toutes les lignes n'ont pas forcément la même longueur.
Maintenant que vous avez compris, je peux vous donner le format de partition :

```
let partition = [| [|note1;freq1|] ; [|note2;freq2|] ... |];;
```

Étape 3 : Lire la partition

Cette dernière étape est sans doute la plus simple à mettre en oeuvre, après tout ce qui a été dit.
L'objectif : parcourir la partition (probablement à l'aide d'une boucle **for**), et jouer toutes les notes rencontrées.

Top Chrono !

Secret (cliquez pour afficher)

Code : OCaml

```
(* Début du code *)

#open "graphics";; (* la commande pour charger graphics *)
```

```
let play = fun part ->    (* La partition en argument *)

 for i=0 to (vect_length part)-1 do    (* On parcourt la
partition *)
   sound (part.(i).(0)) (part.(i).(1))   (* Et on joue la note
(définie par sa fréquence) à la durée donnée *)
 done;;
```

Étape bonus : le tempo

Maintenant que vous êtes en mesure de jouer des notes réparties sur 4 octaves, il serait préférable de pouvoir les jouer pendant une durée prédéfinie en musique, c'est à dire : noire, blanche, croche, demi-croche, ronde...

Or, toutes ces durées dépendent du **tempo**, une valeur généralement comprise entre 40 et 250 qui règle la vitesse à laquelle est jouée la partition. Le tempo correspond en fait au nombre de noires jouées en une minute, et on a donc la formule suivante :

$$noire = \frac{60}{tempo}\ secondes = \frac{60000}{tempo}\ ms$$
$$blanche = 2 \times noire$$
$$croche = \frac{noire}{2}$$

Comme le tempo dépend de la partition jouée, soit vous créez un programme par partition, soit vous créez la partition en fonction du tempo, en définissant la durée des notes comme locale à la partition, ce cette manière :

Code : OCaml

```
let t=70 in  (* TEMPO *)
 let noire = 60000/t and blanche = 120000/t
 and croche = 30000/t and dcroche = 15000/t
 and ronde = 240000/t in
  let partition = [| [|note1;croche|] ; [|note2;noire|] ... |] in
   play partition;;   (* Nécessaire au fonctionnement du code *)
```

Vous êtes devenus de véritables musiciens en Caml, et vous pouvez désormais créer des partitions à votre guise !

Quelques partitions

Je vous ai traduit en CAML deux petites partitions, pas forcément exactes, mais qui feront l'affaire au début, pour que vous les testiez :

Tetris - Song A

Code : OCaml

```
let t=144 in
 let noire = 60000/t and blanche = 120000/t
  and croche = 30000/t and dcroche = 15000/t
  and ronde = 240000/t in
let tetris = [| [|si3;noire|];[|fad3;croche|];[|sol3;croche|];[|la3;noire|];[|sol
[|mi3;noire|];[|mi3;croche|];[|sol3;croche|];[|si3;noire|];[|la3;croche|];[|sol3;
|]

in play tetris;;
```

Requiem for A dream - Main Theme (raccourci)

Code : OCaml

```
let t=70 in
 let noire = 60000/t and blanche = 120000/t
  and croche = 30000/t and dcroche = 15000/t
  and ronde = 240000/t in
   let RFAD =
[|[|sib3;croche|];[|la3;croche|];[|sol3;croche|];[|re3;croche|];[|sib3;croche|];
[|sib3;croche|];[|la3;croche|];[|sol3;croche|];[|re3;croche|];[|sib3;croche|];[|
[|sib4;croche|];[|sol4;noire|];[|sib4;dcroche|];[|sol4;dcroche|];[|sib4;dcroche|
[|sol4;dcroche|];[|sib4;dcroche|];[|sol4;dcroche|];[|sib4;dcroche|];[|la4;noire|
[|sib4;dcroche|];[|la4;dcroche|];[|sib4;dcroche|];[|la4;dcroche|];[|sib4;dcroche
[|la4;dcroche|];[|la4;dcroche|];[|sol4;croche|];[|sib4;dcroche|];[|sib4;dcroche|
[|la4;dcroche|];[|la4;dcroche|];[|sib4;dcroche|];[|sib4;dcroche|];[|sib4;dcroche
[|la4;dcroche|];[|sol4;croche|];[|sib4;dcroche|];[|sib4;dcroche|];[|sib4;dcroche
[|la4;dcroche|];[|sib3;croche|];[|la3;croche|];[|sol3;croche|];[|re3;croche|];[|
[|sib3;croche|];[|sib3;croche|];[|la3;croche|];[|sol3;croche|];[|re3;croche|];[|
[|sib3;croche|];[|sol3;blanche|]|]

in if playing=true then play RFAD;;
```

Have Fun !
J'espère sincèrement que ce tutoriel vous aura plus, et que vous ~~passerez des heures devant Caml à rentrer des partitions~~ pourrez contribuer à ce celui-ci en fournissant les partitions que vous aurez traduit en CAML !