Tutoriel SuperCollider

Chapitre 1

Par Celeste Hutchins

2005

www.celesteh.com

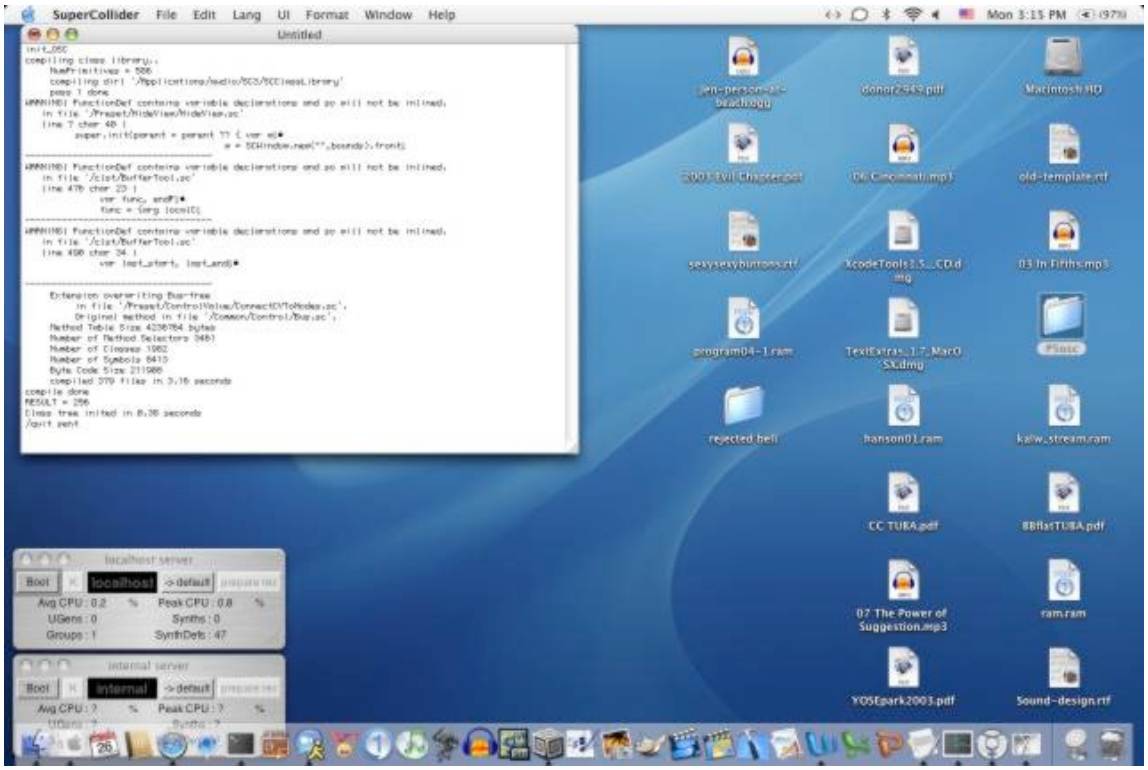
Licence Creative Commons : Attribution uniquement

Chapitre 1 : Introduction

Le but de ce document est de vous enseigner SuperCollider, ainsi que certains concepts de base de la programmation et de la terminologie. Ces concepts sont les mêmes dans de nombreux langages de programmation

Qu'est-ce qu'un **programme** ? Il s'agit d'une série d'instructions qu'un ordinateur suit pour accomplir une tâche. Un programme ressemble beaucoup à une partition. Vous jouez une partition dans l'ordre, de gauche à droite, en jouant chaque note ou repos l'un après l'autre, en sautant en arrière en cas de répétition et en avant en cas de deuxième fin ou de coda. De la même manière, vous pouvez dire à votre ordinateur de jouer un si pendant deux secondes, puis un la, et lui demander de répéter, et ainsi de suite. Vous pouvez demander à votre ordinateur de faire des choses plus compliquées, comme jouer un C si vous déplacez votre souris dans le coin supérieur droit et un D dans la main gauche, ou produire des sons plus compliqués. Pour pouvoir dire toutes ces choses à votre ordinateur, vous devez être capable de parler le même langage que lui. SuperCollider est un langage de programmation conçu pour écrire des programmes musicaux.

Lorsque vous double-cliquez sur l'icône de SuperCollider, trois fenêtres doivent s'ouvrir sur votre écran. Une grande fenêtre de texte appelée "Untitled" doit imprimer des informations et il doit y avoir deux fenêtres plus petites en dessous appelées "localhost server" et "internal server". Si la fenêtre "Sans titre" contient une erreur, les deux fenêtres du serveur ne s'ouvriront pas. Essayez de télécharger une version différente de SuperCollider ou de l'exécuter sur une autre machine.



La fenêtre Sans titre est le lieu de sortie du texte. Les deux autres fenêtres contrôlent deux versions différentes du serveur audio. Les exemples de ce document utilisent le serveur local. Si vous voulez entendre de l'audio, vous devez démarrer le serveur audio, ce que vous pouvez faire en appuyant sur le bouton "Boot". Nous reviendrons plus tard sur ce que cela signifie.

Pour écrire votre propre code, vous devez ouvrir une nouvelle fenêtre, ce que vous pouvez faire dans le menu Fichier ou en tapant pomme-n. Pour exécuter un code, mettez-le en surbrillance à l'aide de la souris, puis appuyez sur la touche Entrée, et NON sur la touche Retour (la touche Entrée peut se trouver à côté des flèches ou sur le pavé numérique). (Pour arrêter un code en cours d'exécution, appuyez sur pomme-point.

Pour obtenir de l'aide, appuyez sur apple-shift- ? Pour obtenir de l'aide sur un sujet spécifique, par exemple sur SynthDef, sélectionnez le mot SynthDef et appuyez sur apple-shift- ?

Dans Supercollider, les blocs de code sont placés entre parenthèses. Pour exécuter un bloc entier, double-cliquez à droite de la parenthèse ouverte et appuyez sur Entrée. Par exemple, si vous double-cliquez à droite de la parenthèse ouverte de :

```
(
  "hello world".postln ;
)
```

alors `hello world` s'imprimera dans la fenêtre Untitled.

Objets

SuperCollider est un **langage orienté objet**. `hello world"` est un type d'**objet** appelé "**chaîne"**. Une chaîne est un morceau de texte entouré de guillemets doubles. Il existe d'autres objets, par exemple les entiers, les nombres à virgule flottante ou les tableaux, dont nous parlerons plus tard. Nous pouvons communiquer avec les objets en utilisant les **méthodes** définies par l'auteur de l'objet. Dans l'exemple ci-dessus, nous envoyons un **message** de type `postln` à la chaîne de caractères. Ce message demande à la chaîne de s'imprimer elle-même.

Nous pouvons prendre l'exemple d'un interrupteur. Pour allumer une lumière, il suffit d'appuyer sur l'interrupteur. Cela revient à envoyer un message à l'ampoule : "Hé, allume". Si l'on baisse l'interrupteur, c'est comme si l'on envoyait un message du type "hé, éteins".

Variables

Considérons une personne, Nicole, comme un objet. Nicole est elle-même un objet, mais le mot "Nicole" est un nom qui se réfère à Nicole la personne. De la même manière, nous pouvons donner des noms à nos objets. Ces noms sont

appelés **variables**.

```
(
  var greeting ;

  greeting = "hello world" ;
  greeting.postln ;
)
```

Que se passe-t-il ici ? Le premier mot, "var", est l'abréviation de **variable**. Une variable est un emplacement de stockage. C'est un endroit où l'on peut stocker des données. Le contenu des données stockées peut varier, c'est pourquoi on parle de variable. Le deuxième mot "greeting" est un nom. C'est le nom des variables. Ici, nous **déclarons** à l'**interprète** que nous aurons une variable nommée "greeting". L'interpréteur est la partie de SuperCollider qui lit et exécute nos programmes. Ainsi, lorsque l'interpréteur lit notre programme et voit le mot "greeting", il sait que greeting est une variable que nous avons créée. Sinon, il ne saurait pas de quoi nous parlons.
Tous les noms de variables dans SuperCollider doivent commencer par une lettre minuscule et ne peuvent pas être des **mots réservés**. Vous ne pouvez pas avoir une variable appelée "var" car var a déjà une signification particulière pour l'interpréteur.

Ensuite, nous **assignons** une valeur à la variable. `greeting` obtient `"hello world"`. La variable se trouve à gauche du signe égal. Elle doit toujours être à gauche. Il ne peut y avoir qu'une seule chose à gauche d'un signe égal. Ce signe dit, hé, prenez ce qui se trouve à droite de ce signe égal et stockez-le sous le nom de la variable à gauche.

Dans la dernière ligne, nous envoyons un message `postln` à la variable. L'interpréteur SuperCollider envoie ce message à `greeting`. `greeting` est une chaîne de caractères. Les chaînes s'impriment elles-mêmes avec le message postln. Donc, comme `greeting` est une

Chaîne, le contenu du `message d'accueil`, `"hello world"`, s'imprime.
À gauche se trouve le nom de la variable, un objet. Vient ensuite un point. Vient ensuite le nom du message. Il existe plusieurs styles de codage possibles dans SuperCollider, mais nous allons nous concentrer sur la **notation du récepteur** parce qu'elle est commune à de nombreux langages de programmation. C'est-à-dire

```
objet.message ;
```

Remarquez que chaque ligne se termine par un point-virgule. Dans Supercollider, toutes les lignes doivent **se terminer par un** point-virgule. Plus tard, lorsque vous essayerez de comprendre pourquoi un programme ne fonctionne pas, vous découvrirez qu'il manque un point-virgule. Le point-virgule indique à l'interpréteur que vous en avez terminé avec cette instruction. Une instruction peut s'étendre sur autant de lignes que nécessaire, mais elle doit se terminer par un point-virgule.

Classes

Produisons du son. Tout d'abord, assurez-vous de démarrer le serveur local. Le serveur local est l'une des deux petites boîtes grises en bas à gauche de votre écran. Appuyez sur le bouton "Boot". Lorsque le serveur est démarré, le mot "localhost" devient rouge et le bouton "Boot" se transforme en "Quit".

Ensuite, sélectionnez ce code et appuyez sur la touche Entrée (et non Retour !). Vous pouvez le sélectionner en double-cliquant à droite de la parenthèse ouverte.

(

```
	var syn, sound ;

	syn = SynthDef.new("example1", {
		Out.ar(0, SinOsc.ar(440)) ;
	}) ;

	syn.load(s) ;

	sound = Synth.new("example1") ;
)
```

Lorsque vous voulez arrêter le programme, appuyez sur apple-period.

Que se passe-t-il ici ? Les programmes de SuperCollider sont **interprétés**. Cela signifie que vous devez les exécuter à partir de l'**interpréteur de** SuperCollider. Vous n'obtiendrez pas une petite icône sur laquelle vous pouvez double-cliquer, comme c'est le cas pour Microsoft Word. Lorsque vous mettez du texte en surbrillance dans SuperCollider et que vous appuyez sur Entrée, vous demandez à SuperCollider de l'exécuter en tant que programme.

Notre programme déclare d'abord une variable, puis il définit un SynthDef et le charge sur le serveur, avant de créer un objet Synth qui joue le son. C'est un exemple compliqué, mais je voulais que vous produisiez des sons le plus rapidement possible. Examinons l'exemple ligne par ligne.

La première ligne de notre programme est une parenthèse ouverte. Les **blocs** de **code** sont entourés de parenthèses dans SuperCollider.

La ligne suivante est `var syn, sound ;`. Nous déclarons deux variables. L'une s'appelle syn et l'autre s'appelle sound.

La ligne suivante, traduite en anglais, signifie "Je veux créer un nouveau **SynthDef** appelé 'exemple1' et le stocker dans la variable syn". Whoa, qu'est-ce qu'un SynthDef ? Le fichier d'aide de SynthDef me dit qu'un SynthDef est une "définition de l'architecture d'un synthé". J'ai accédé au fichier d'aide en surlignant le mot SynthDef et en appuyant sur apple-shift- ? Lorsque vous voyez un terme que vous ne comprenez pas, vous pouvez accéder au fichier d'aide (s'il existe) en sélectionnant le terme et en appuyant sur apple-shift- ?

SuperCollider n'est pas un seul programme, mais deux. Une partie du programme est l'interprète, dont nous avons parlé jusqu'à présent. L'autre partie du programme est le **serveur audio**, qui produit le son. Le serveur audio s'exécute séparément de l'interprète, mais ils peuvent communiquer entre eux à l'aide de ce que l'on appelle l'**OSC**, dont nous parlerons plus tard.

Le serveur ne sait rien d'autre que ce qu'il doit savoir. Cela lui permet de fonctionner beaucoup plus rapidement et efficacement. Vous pouvez lui demander de produire certains types de sons, mais pour qu'il fonctionne rapidement, il doit savoir quels types de sons vous voulez avant que vous ne les produisiez. Vous écrivez une description de ce que vous voulez que le serveur fasse. Cette description, ou définition, est appelée SynthDef.

`SynthDef` est le nom d'une **classe**. Les définitions d'objets sont appelées classes. Un objet est une **instance** particulière d'une classe. Dans SuperCollider, tous les noms de classes commencent par une majuscule.

Exemples : Nicole est un exemple de personne. Rappelez-vous que Nicole est un objet. "Personne" serait sa classe. De même, SynthDef est une classe.

Il existe certains types de messages que vous pouvez envoyer à une classe. L'un d'entre eux s'appelle un **constructeur**. Le message ".new" est un constructeur. Il indique à la classe de créer une nouvelle instance d'elle-même. Ainsi, lorsque nous appelons SynthDef.new, nous obtenons en retour un

nouveau SynthDef, que nous stockons dans la variable syn.

À l'intérieur des parenthèses se trouvent des **arguments**. Certains messages, comme postln, savent ce qu'ils doivent faire lorsqu'ils sont appelés - si nous plaçons postln après une chaîne de caractères comme nous l'avons fait ci-dessus avec "hello world".postln, le message sait simplement qu'il doit imprimer la chaîne de caractères qu'il contient. D'autres messages nécessitent plus d'informations, que nous appelons **arguments**. Remarquez que nous avons un objet, un message et des arguments. La façon dont nous les codons est la suivante :

```
object.message(argument1, argument2, ... argumentN) ;
```

Ou

```
Class.message(argument1, argument2, ... argumentN) ;
```

Dans le cas de notre SynthDef, `SynthDef` est la classe, `new` est le message. Et `"exemple1"` et ce qui se trouve entre les crochets sont les deux arguments.

Les éléments entre crochets indiquent au SynthDef ce qu'il doit jouer. Les éléments entre crochets {} sont appelés des **fonctions**. Une fonction est un type spécial d'objet qui est composé d'un bloc de code que vous pouvez exécuter. Nous y reviendrons plus tard.

Out est un **UGen**. Le fichier d'aide pour les UGens dit, "Un UGen ou **générateur d'unité** est un objet pour générer ou traiter des signaux audio ou de contrôle." Les UGen existent dans les SynthDefs, ils constituent la base d'un SynthDef. Out est un UGen qui écrit un signal sur un **bus**, qui, dans ce cas, envoie sa sortie sur le canal gauche. .ar est un type de constructeur. Out.ar crée donc une nouvelle instance d'un UGen Out fonctionnant au **taux audio**. ar signifie "taux audio" et est un nom de constructeur courant pour les UGen.

En regardant les arguments de Out.ar, 0 signifie canal gauche. Si c'était un 1, cela signifierait canal droit. L'élément suivant est ce qui est envoyé, c'est-à-dire

un générateur de tonalité sinusoïdale. L'argument de SinOsc.ar, 440, est la fréquence à jouer.

Ainsi, SinOsc.ar(440) crée une onde sinusoïdale à 440 Hz. Out.ar prend cette onde sinusoïdale et l'envoie au canal gauche, qui est le canal 0.

La ligne suivante est constituée d'une parenthèse fermée, d'une parène fermée et d'un point-virgule. C'est la fin de la fonction, la fin du SynthDef et la fin d'une déclaration. Nous avons créé un nouveau SynthDef qui inclut une fonction décrivant ce que le SynthDef doit faire lorsqu'il est instancié.

La ligne suivante dit de prendre ce SynthDef et de le charger sur le serveur. Nous envoyons un message à syn, disant load. L'argument, s, est le serveur auquel l'envoyer. Dans SuperCollider, un s minuscule seul fait référence au serveur audio.

La ligne suivante demande de créer un nouveau Synth. Un Synth est une instance d'une SynthDef. L'interpréteur envoie un message au serveur, disant, "Hey, pourrais-tu faire un synthé qui joue le SynthDef appelé 'exemple1' ?" Le serveur regarde et dit, "oh oui, j'ai un SynthDef appelé ça" et crée une nouvelle instance d'un Synth, qui exécute la fonction que nous avons définie.

Nous obtenons ainsi un nouvel objet Synth et le stockons dans "sound".

Disons que nous ne voulons pas jouer un la. Disons que nous voulons jouer un mi. Nous pouvons remplacer 440 par 660.

```
(
	var sdef ;
	sdef = SynthDef.new("example1", {

		Out.ar(0, SinOsc.ar(660)) ;
	}) ;

	sdef.play ;
```

```
)
```

Mais lorsque nous écrivons un morceau, nous ne voulons pas avoir à écrire un nouveau SynthDef pour chaque note que nous allons jouer. Nous pouvons créer notre propre argument, qui indiquera à la SynthDef quelle fréquence jouer.

```
(

	var syn, sound ;

	syn = SynthDef.new("example1", { arg freq ;
		Out.ar(0, SinOsc.ar(freq)) ;
	}) ;

	syn.load(s) ;

	sound = Synth.new("example1", [\freq, 440]) ;
)
```

Nous appelons cet argument "freq". Un argument est un type particulier de variable. Vous les déclarez au début d'un bloc de code en utilisant le mot réservé "arg". Ainsi, la partie `"arg freq ;"` de `"syn = SynthDef.new("example1", { arg freq ;"` indique à SuperCollider que notre fonction SynthDef prend un seul argument appelé freq. Nous pouvons alors utiliser freq comme n'importe quelle autre variable. Ici, nous passons à SinOsc.ar, à utiliser pour la fréquence.

Le passage de variables aux Synths est un peu différent du passage normal de variables. L'interprète doit communiquer avec le serveur audio. Il utilise un protocole appelé OSC. L'objet Synth va gérer l'OSC pour vous, mais le serveur ne sait pas à quel argument vous essayez de passer une valeur à moins que vous ne le lui disiez spécifiquement. Synth.new prend donc un second argument optionnel, qui est un **tableau**. Un

est une liste séparée par des virgules et entourée de crochets. Lorsque vous transmettez des arguments au serveur via un synthétiseur, le tableau doit être constitué de paires. Le premier élément d'une paire est un **symbole** ou une chaîne et le second est une valeur. Le symbole ou la chaîne doit correspondre au nom de l'argument. Ainsi, pour un argument appelé freq, nous utilisons le symbole \freq ou la chaîne "freq". Si nous avions un argument appelé foo, nous utiliserions le symbole \foo ou la chaîne "foo". Un symbole commence toujours par une barre oblique : "\". Ainsi, pour passer une valeur de 440 à l'argument freq, notre tableau contient [\freq, 440]. Si nous avions deux arguments, l'un freq et l'autre foo, nous pourrions passer des valeurs à freq et foo avec un tableau qui ressemblerait à [\freq, 440, \foo 647]. Le tableau est composé de paires, le 440 va à freq et le 647 à foo. Les symboles ou chaînes doivent correspondre aux noms des arguments pris par la fonction SynthDef.

Problèmes

Lorsque vous écrivez du code, si vous voulez le colorer syntaxiquement, de sorte que les mots réservés soient en bleu, les chaînes en gris, les symboles en vert, et ainsi de suite, vous pouvez trouver cela dans le menu Format ou simplement taper "apple-" (apple - guillemets simples).

1. Écrivez votre propre version de "hello world".

2. Ecrivez vos propres SynthDefs, en utilisant certains des UGens d'oscillateurs. Pour en trouver la liste, surlignez le mot "UGens" et tapez apple-shift- ?. Quelques oscillateurs à essayer sont Saw et Pulse.