Chapter 1

Intro

Dans cet article je vais vous présenter une facette de Pharo qu'on a pas l'habitude de voir, un domaine pour lequel il peut surprendre. Ce domaine c'est la synthèse sonore. Dès qu'on a besoin de sonoriser un peu notre tout nouveau programme on pense soit à du mp3 pour la musique ou des sons digitalisés pour les bruitages. Dans Pharo on dispose d'outils et bibliothèques permettant créer de toute pièce du son synthétisé pour nos production. Nous allons vous montrer comment on fait cela en quelques ligne de code seulement.

Tout d'abord récupérer et installer la version XXX de Pharo à l'adresse: http://www.pharo.org/YYY

rajouter l'adresse de téléchargement

Chapter 2

Bases de la synthèse FM par l'exemple

Avant de se familiariser avec les différentes méthodes utiles on va s'assurer que notre environnement puisse jouer un son convenablement. Pour cela on va ouvrir un workspace (depuis le menu Tools) et évaluer la ligne de code suivante. Nous vous expliquerons par la suite ce dont il s'agit.

```
FMSound organ1 play.
```

Si aucun son n'est produit vous pouvez vérifier ces premiers points

Indiquer la marche à suivre ou une page internet l'indiquant ?

- Le volume sonore
- Le plugin sonore n'est pas installé.
- Le process sonore ne fonctionne plus. Il faut l'arrêter puis le relancer en jouant

```
SoundPlayer stopPlayerProcess
SoundPlayer startUp
```

Maintenant que le son est restauré nous allons voir comment celui ci est généré, comment nous pouvons créer le nôtre. Si on regarde de plus près le code de la méthode de classes qui définit l'instrument **organ1** on voit ceci :

```
FMSound class>>organ1
   "FMSound organ1 play"
   "(FMSound majorScaleOn: FMSound organ1) play"
```

```
| sound |
sound := self new.
sound addEnvelope: (VolumeEnvelope
                    points: { 0@0 . 60@1.0 . 110@0.8 . 200@1.0 . 250@0.0}
                    loopStart: 2 loopEnd: 4).
sound setPitch: 440.0 dur: 1.0 loudness: 0.5.
   ^ snd
```

Regardons étape par étape comment on crée un son. Tout d'abord crée une nouvelle instance de la classe **FMSound**

```
snd := FMSound new.
```

On définit ensuite un ensemble de points dans une collection qui vont nous servir juste après pour l'enveloppe du volume. Pour chaque point la première donnée est un nombre en miliseconde, la seconde est le volume exprimé de 0 à 1 (1 étant égal à 100% du volume). Notez qu'utiliser un point est un abus et que le programmeur aurait du utiliser une paire.

```
p := OrderedCollection new.
p add: 0@0; add: 60@1.0; add: 110@0.8; add: 200@1.0; add: 250@0.0.
```

la phrase ne va pas du tout, a reformuler

L'enveloppe est définie d'après un ensemble de points et d'une boucle. Elle peut aussi bien reprendre l'ensemble des points ou seulement boucler sur 1 seul point. On peut ainsi jouer avec le volume d'un son pour qu'il s'atténue, ou pour donner un effet d'écho par exemple. On ajoute cette enveloppe au son avec une boucle sur l'indice 2 et 4. Si on regarde les valeurs on voit que ça oscille entre 0.8 et 1.

```
snd addEnvelope: (VolumeEnvelope points: p loopStart: 2 loopEnd: 4).
```

La méthode retourne le son avec comme paramètre la note (ici exprimée en Hz, voir tableau **10.1**), la durée en seconde et un volume global de la note. Ce volume sera ensuite modulé grâce à l'enveloppe qui est définie.

```
^ snd setPitch: 440.0 dur: 1.0 loudness: 0.5
```

Chapter 3

Comment faire pour jouer notre propre son ?

Nous pouvons jouer un LA comme suit:

```
(FMSound new setPitch: 440 dur: 1.0 loudness: 1) play
```

Le son est joué à une fréquence de 440 Hz pendant 1 seconde.

La classe **FMSound** permet la génération de sons sinusoïdaux (voir schéma). Ça donne un son assez doux et régulier.

Dans ces exemples on a défini une note avec sa fréquence. Si on veut jouer une seule note l'écriture est simple, mais cela peut rapidement devenir long, fastidieux et les erreurs difficiles à détecter si on doit écrire tout une mélodie de cette manière.

Une chose à savoir est qu'au lieu d'écrire une note définie par sa fréquence on peut l'écrire dans sa notation anglaise. -> Voir tableau.

Le LA de tout à l'heure peut s'écrire maintenant comme cela

```
(FMSound new setPitch: 'a4' dur: 1.0 loudness: 1) play
```

Le sol dièse par exemple s'écrit

```
(FMSound new setPitch: 'g#4' dur:1.0 loudness: 1) play
```

Chapter 4

Jouer plusieurs sons en séquence

Jouer un son c'est bien, mais en jouer deux c'est mieux. Pour cela nous avons à notre disposition une méthode nommée **SequentialSound**. Comme son nom l'indique on va pouvoir jouer à la suite chacun des sons qu'on souhaite.

```
|seq|
seq := SequentialSound new.
seq add: ((FMSound new) setPitch: 'g#4' dur:0.5 loudness: 1).
seq add: ((FMSound new) setPitch: 'a3' dur:0.5 loudness: 1).
seq add: ((FMSound new) setPitch: 'g#4' dur:0.5 loudness: 1).
seq play.
```

On joue 3 notes les unes à la suite des autres. C'est mieux mais ce n'est toujours pas très intéressant.La première chose dont on se rend compte c'est le petit clap à la fin de chacune des notes. Si on veut éviter cela on va rajouter à chaque note une enveloppe de volume.

Le code va changer un peu

```
|seq snd1 snd2 snd3 points|
"Declaration des sons"
seq := SequentialSound new.
snd1 := FMSound new.
snd2 := FMSound new.
snd3 := FMSound new.
points := OrderedCollection new.

"Creation du volume pour les instruments"
points add: 0@1;add: 400@0.4;add: 500@0.

"Generation des notes"
```

```
snd1 setPitch: 'g#4' dur:0.5 loudness: 1.
snd2 setPitch: 'a3' dur:0.5 loudness: 1.
snd3 setPitch: 'g#4' dur:0.5 loudness: 1.

"Application de l'enveloppe aux sons"
snd1 addEnvelope: (VolumeEnvelope points: points loopStart: 3  loopEnd:3 ).
snd2 addEnvelope: (VolumeEnvelope points: points loopStart: 3  loopEnd:3 ).
snd3 addEnvelope: (VolumeEnvelope points: points loopStart: 3  loopEnd:3 ).

"Les sons les uns a la suite des autres"
seq add: snd1.
seq add: snd2.
seq add: snd3.

"On joute le tout"
seq play.
```

Ça a tout de suite un peu plus de tenue non ? la séquence est plus agréable à écouter et les claps de fin ne sont plus audibles.

Avec un peu créativité on va pouvoir faire quelque chose de ce son.

Chapter 5

Jouer plusieurs sons simultanément

Jouer des sons en séquence c'est bien mais pas encore suffisant pour donner une âme à notre production. Si on veut avoir quelque chose qui sonne correctement à nos oreilles il va falloir maintenant rajouter ce qu'on va appeler des pistes. La classe **MixedSound** va nous aider à cela.

```
|mix snd1 snd2 points|
mix := MixedSound new.
snd1 := FMSound new.
snd2 := FMSound new.

points := OrderedCollection new.

"Creation du volume pour les instruments"
points add: 0@1;add: 400@0.4;add: 500@0.

"Generation des notes"
snd1 setPitch: 'a4' dur:1 loudness: 1.
snd2 setPitch: 'a3' dur:1 loudness: 1.

"Application de l'enveloppe aux sons"
snd1 addEnvelope: (VolumeEnvelope points: points loopStart: 3  loopEnd:3 ).
snd2 addEnvelope: (VolumeEnvelope points: points loopStart: 3  loopEnd:3 ).

mix add: snd1 pan: 0; add: snd2 pan: 1.

mix play.
```

Le son généré ressemble un peu aux ceux qu'on pouvait entendre sur les jouets pour enfants. Modifiez un peu les valeurs **setPitch** pour obtenir des

notes différentes et vous familiariser avec les méthodes. Les enveloppes de volume modifiant la courbe du son au cours du temps peuvent être également modifiées pour obtenir des effets particuliers.

Remplacez les valeurs des points par

```
points add: 0@1;add: 100@0.6; add: 200@0; add: 300@0.6;add: 400@0.2;add: 500@0.
```

et la déclaration des enveloppes par

```
snd1 addEnvelope: (VolumeEnvelope points: points loopStart: 6  loopEnd:6 ).
snd2 addEnvelope: (VolumeEnvelope points: points loopStart: 6  loopEnd:6 ).
```

Le son est comme doublé. Le premier étant plus fort que l second. La variation entre les est douce et continue. Un calcul est fait entre 2 valeurs de volume pour éviter les cassures pas jolies à l'oreille. On peut cependant le faire en ayant un temps très court entre 2 valeurs.

```
points add: 0@1;add: 100@0.6; add: 200@0; add: 201@0.6;add: 400@0.2;add: 500@0.
```

La cassure est ici nette.

Si vous regardez bien un paramètre s'est glissé lors de l'appel à **add:**. Le paramètre **pan:** va positionner le son sur le canal stéréo. **0** pour gauche, **1** pour droite. Une valeur entre les 2 limites va déplacer le son vers le milieu. **0.5** étant le centre. Jouez un peu avec les valeurs pour séparer les 2 sons.

Chapter 6

Les enveloppes

L'enveloppe modifie le son au cours du temps. On a vu qu'on pouvait changer le volume, mais on peut également faire varier la note pendant qu'elle est jouée, c'est ce qu'on appel le pitch. De la même manière que pour le volume on va définir un ensemble de points qui vont indiquer quand changer la fréquence du son. Plus la valeur est haute, plus la note sera haute. Si on indique un chiffre négatif, la note descendra en dessous de la note de départ.

```
|snd p|
p := OrderedCollection new.
p add: 0@0; add: 100@0.1; add: 200@0.2; add: 300@(−0.5);
  add: 400@0.7; add: 800@0.8; add: 1000@1.
snd := FMSound new.
snd addEnvelope: (PitchEnvelope points: p loopStart:7  loopEnd:7 ).
(snd setPitch: 'a4' dur: 2 loudness: 0.5)play.
```

La variation se fait en douceur, sans cassures. En couplant cela à une enveloppe de volume on peut commencer à produire un son intéressant.

```
|snd p p2|
p := OrderedCollection new.
p2 := OrderedCollection new.
p add: 0@0; add: 100@0.1; add: 200@0.2; add: 300@(−0.5);
  add: 400@0.7; add: 800@0.8; add: 1000@1.
p2 add: 0@0; add: 100@0.1; add: 200@0.2; add: 300@(−0.5);
   add: 400@0.7; add: 800@0.8; add: 1000@0.
snd := FMSound new.
snd addEnvelope: (PitchEnvelope points: p loopStart:7  loopEnd:7 ).
snd addEnvelope: (VolumeEnvelope points: p2 loopStart:7  loopEnd:7 ).
(snd setPitch: 'a4' dur: 2 loudness: 0.5)play.
```

La définitions des points pour les enveloppes peuvent utiliser une méthode appelée **exponentialDecay:** de la classe **VolumeEnvelope** pour le volume ou **PitchEnvelope** pour le pitch. La courbe va être de type exponentielle partant du volume le plus haut pour arriver en douceur à 0.

La définition du volume devient alors ceci

```
snd addEnvelope: (VolumeEnvelope exponentialDecay: 0.9).
```

La valeur doit être plus grande que 0 et plus petite que 1.Pour connaître l'ensemble des points générés il suffit de faire un **clic droit**/**Print it** sur la ligne de code suivante

```
(VolumeEnvelope exponentialDecay: 0.2) points.
```

Le résultat est le suivant

```
{(0@0.0). (10@1.0). (20@0.2). (30@0.04000000000000001). (50@0.0)}
```

Nous avons donc vu comment sonorisé un peu nos productions. Pour le moment ça se limite à un son, plusieurs sons joués les uns après les autres ou encore un plusieurs sons joués en même temps.

On a entre nos mains pratiquement tous les outils dont on a besoin pour écrire notre première mélodie, que dis-je, notre première super production !

Chapter 7

Jouer une mélodie

A quoi peut bien ressembler de la musique produite dans Pharo ? Evaluons la ligne de code suivante dans un workspace et écoutons.

```
FMSound bachFugue play
```

La musique est une séquence de notes. Nous savons maintenant comment écrire une telle séquence mais pour une mélodie entière ce n'est pas adapté. Il est nécessaire de faire autrement.

Nous allons décrire l'ensemble des notes dans un tableau. Le format est le suivant

```
(note duree volume)
```

La note peut être écrite sous sa forme littérale ou en hertz. La durée s'exprime en millisecondes. Le volume quant à lui est une valeur entre 0 et 1000. Au delà de 1000 le son peut saturer mais cela peut se révéler utile dans certain cas pour donner un effet.

Par exemple

```
notes := #((a3 200 150) (a4 100 150) (g3 150 150))
```

Je vais vous aider à écrire notre premier morceau avec un air qui devrait rappeler des bons souvenirs à tout ceux qui ont connu le début des jeux vidéos

Nous allons dans un premier temps déclarer la classe permettant de mixer l'ensemble des pistes.

```
FMSound class>>pacman
   "comment stating purpose of message"

   ^ MixedSound new
```

```
    add: (self pacmanV1On: self fm1) pan: 0.5;
    add: (self pacmanV2On: self fm2) pan: 0.5.
```

Les 2 pistes suivantes contiennent l'ensemble des notes à jouer avec la note, la durée et le volume.

```
FMSound class>>pacmanV1On: aSound

  |notes|
  notes := #(
    (b4 0.144 150) (b5 0.144 150) ('f#5' 0.144 150) ('d#5' 0.144 150)
    (b5 0.072 150) ('f#5' 0.144 150) ('d#5' 0.288 150) (c5 0.144 150)
    (c6 0.144 150) (g5 0.144 150) (e5 0.144 150) (c6 0.072 150)
    (g5 0.216 150) (e5 0.288 150) (b4 0.144 150) (b5 0.144 150)
    ('f#5' 0.144 150) ('d#5' 0.144 150) (b5 0.072 150) ('f#5' 0.216 150)
    ('d#5' 0.288 150) ('d#5' 0.072 150) (e5 0.072 150) (f5 0.144 150)
    (f5 0.072 150) ('f#5' 0.072 150) (g5 0.144 150) (g5 0.072 150)
    ('g#5' 0.072 150) (a5 0.144 150) (b5 0.216 150)).

    ^self noteSequenceOn: aSound from: notes.
```

Lorsqu'un son sinusoïdal possède une fréquence trop basse, celui ci est difficilement audible. Le volume va être remonté sur cette piste pour une meilleure écoute.

```
FMSound class>>pacmanV2On: aSound
  |notes|
  notes := #(
    (b1 0.432 500) (b2 0.144 500) (b1 0.432 500) (b2 0.144 500) (c2 0.432 500)
    (c3 0.144 500) (c2 0.432 500) (c3 0.144 500) (b1 0.432 500) (b2 0.144 500)
    (b1 0.432 500) (b2 0.144 500) ('f#2' 0.288 500) ('g#2' 0.288 500)
    ('a#2' 0.288 500) (b2 0.288 500)).

    ^self noteSequenceOn: aSound from: notes.
```

Et maintenant nos 2 instruments que l'on va utiliser

```
FMSound class>>fm1
  |snd|

  snd := self new.

  snd addEnvelope: (VolumeEnvelope
        points: { 0@1 . 300@0 }
        loopStart: 2
        loopEnd: 2
        ).
  snd modulation: 1 ratio: 2.
```

```
    ^snd
```

```
FMSound class>>fm2
    |snd|

    snd := self new.

    snd addEnvelope: (VolumeEnvelope
            points: {  0@1 . 300@0 }
            loopStart: 2
            loopEnd: 2
            ).
    snd modulation: 1 ratio: 2.

    ^snd
```

Les 2 classes définissant les instruments sont rigoureusement les mêmes, mais je vous invite une fois de plus à modifier les valeurs de l'enveloppe ou de la modulation et d'écouter le résultat.

Chapter 8

Jouer un sample à partir d'un fichier ou directement depuis la mémoire

La sonorisation d'une production ne passe pas essentiellement par de la synthèse sonore, parfois il est nécessaire de faire appel à des bruitages ou des voix enregistrées.

Essayons dans un premier temps de jouer un son à partir d'un fichier au format wav

```
(SampledSound fromWaveFileNamed: '/home/messner/sound.wav') play
```

Le son est joué directement depuis le fichier, cependant si nous faisions un jeu, les chargements risqueraient de saccader les animations et rendre le jeu injouable. Afin d'éviter ces petits désagréments les sons seront chargés en mémoire et joués lorsque ce sera nécessaire.

Pour cela on va le rajouter à notre librairie de sons

```
|spl spl2|

"Chargement du son a partir d'un fichier"
spl := SampledSound fromWaveFileNamed: '/home/messner/guitar.wav'.
"Ajout du son dans la librairie pour le jouer plus tard"
SampledSound addLibrarySoundNamed: 'guitar'
        samples: spl samples
        samplingRate: spl originalSamplingRate.
```

le "à" n'est pas correctement interprété dans un bloc. Pkoi ???

Il nous reste plus qu'à le jouer à la fréquence et à la durée souhaitée

```
|spl spl2|
spl := SampledSound soundNamed: 'guitar'.
spl2 := (SampledSound samples: spl samples samplingRate: 2000).
spl2 duration: 0.1.
spl2 play.
```

Chapter 9

Un dernier petit effort

Nous allons clore cet article sur une dernière méthode permettant de jouer plusieurs sons digitalisés en séquence.

Dans un premier temps nous allons charger en mémoire les différents samples qui seront utilisés.

```
|spl spl1 spl2|

"Chargement des sons"
spl := SampledSound fromWaveFileNamed: '/home/messner/ocean.wav'.
spl1 := SampledSound fromWaveFileNamed: '/home/messner/mouette.wav'.
spl2 := SampledSound fromWaveFileNamed: '/home/messner/bateau.wav'.

"Ajout des sons dans la librairie pour les jouer plus tard"
SampledSound addLibrarySoundNamed: 'ocean'
        samples: spl samples
        samplingRate: spl originalSamplingRate.
SampledSound addLibrarySoundNamed: 'mouette'
        samples: spl1 samples
        samplingRate: spl1 originalSamplingRate.
SampledSound addLibrarySoundNamed: 'bateau'
        samples: spl2 samples
        samplingRate: spl2 originalSamplingRate.
```

Maintenant que tout nos sons sont chargés nous allons les jouer en séquence grâce à la méthode **QueueSound**.

```
|mix q1 q2 spl1 spl2 spl3|

q1 := QueueSound new.
q2 := QueueSound new.
mix := MixedSound new.
spl1 := (SampledSound samples: (SampledSound soundNamed: 'ocean')
```

```
                samples samplingRate: 22000).
spl1 duration: 20.
spl2 := (SampledSound samples: (SampledSound soundNamed: 'mouette')
                samples samplingRate: 22000).
spl2 duration: 10.
spl3 := (SampledSound samples: (SampledSound soundNamed: 'bateau')
                samples samplingRate: 22000).
spl3 duration: 5.

q1 add:spl2;add:spl2.
q2 add:spl3;add:spl3.
mix add: q1 pan: 0 volume: 0.100;
   add: q2 pan: 1 volume: 0.5;
   add: spl1 pan: 0.5 volume: 1.
mix play.
```

Chapter 10

Annexes

10.1 Tableau des fréquences

Note	Octave								
	-1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
C	16.3	32.7	65	131	262	523	1046.5	2093	4186
C#	17.3	34.6	69	139	277	554	1109	2217	4435
D	18.3	36.7	74	147	294	587	1175	2349	4698
D#	19.4	38.9	78	156	311	622	1244.5	2489	4978
E	20.5	41.2	83	165	330	659	1318.5	2637	5274
F	21.8	43.6	87	175	349	698.5	1397	2794	5588
F#	23.1	46.2	92.5	185	370	740	1480	2960	5920
G	24.5	49.0	98	196	392	784	1568	3136	6272
G#	26.0	51.9	104	208	415	831	1661	3322	6645
A	27.5	55.0	110	220	440	880	1760	3520	7040
A#	29.1	58.0	117	233	466	932	1865	3729	7458
B	30.8	62.0	123	247	494	988	1975	3951	7902