Projet traitement de signal :
Réalisation d'une boîte à effets sonores

ING 3 - TD9

Vincent Kocupyr

Léopold Greaujean

Youen Le Cloirec

Sommaire

I. Présentation du Projet

Le but du projet est de réaliser sous Matlab des effets sonores et une interface graphique nommée GUI (Graphical User Interface) permettant à l'utilisateur, à l'aide de différents « push butons », de modifier des sons pré-enregistrés en leur appliquant ces différents effets sonores.

Il était demandé pour ce projet de réaliser trois classes d'effets sonores :

- Effet temporel
- Effet fréquentielle
- Effet Dynamique

Afin d'agrémenter le projet, l'interface graphique ce devait d'être simple d'utilisation, mais contenant assez de manipulation afin de pouvoir "s'amuser" à sa guise avec les différents sons.
Ainsi nous est venu l'idée de mettre à disposition de l'utilisateur non pas un son, mais deux sons. Pouvant être similaire ou différent, écouter en même temps, l'un après l'autre, ou bien même de manière séparé (un son dans l'écouteur gauche, et l'autre dans l'écouteur droit)

II. Le Code

1) Variables

Pour ne pas se perdre dans la programmation du logiciel, nous avons créer deux objets correspondant aux objet de musiques, et quelques variable qui sont nécéssaire au fonctionnement du programme :

*Objet Musique (obj1 est la musique de gauche, obj2 la musique de droite)

handles.obj1x -->	x est le signal audio échantilloné récupéré par wavread
handles.obj1fo -->	fo est la fréquence original du signal récupéré par wavread
handles.obj1fe -->	fe est la fréquence du signal utiliser traiter dans les effets
handles.obj1n -->	n est le nombre de bit du signal récupéré par wavread
handles.obj1audio -->	audio est l'objet audioplayer obtenue avec par audioplayer(x1,Fe1);
handles.obj1aff -->	aff est le signal traité par l'effet, utilisé pour le graphique
handles.obj1tmp -->	tmp est le temps du signal, utilisé pour le graphique
handles.obj1pauseid -->	pauseid est un booléen indiquant si l'écoute est en pause
handles.obj1acc -->	l'indice d'accélération (de 1 à 2 soit 2 fois plus vite)
handles.obj1dec -->	l'indice de décéleration (de 1 à 0,5 soit 2 fois plus lent)
handles.obj1speed -->	booléen indiquant si le son doit être accéléré ou décéléré

*Variable communes

handles.selectmusique -->	booléen indiquant la musique selectionné pour l'effet choisi
handles.separatemode -->	booléen indiquant si le mode séparé est activé ou non

2) scripts et fonction

Afin de pouvoir travailler plus facilement sur les différents effet, le premier objectif une fois l'utilisation de matlab et de l'interface GUI assimilé, fut de mettre en place quelques script et fonction ayant leur utilité dans le but de faciliter la programmation des effets, et éviter la redondance de code dans ces derniers.

* Channel1

```
function x1 = channel1(x)
% Renvoie la première colonne de x
    x1 = x(:,1);
end
```

* Channel2

```
function x2 = channel2(x)
% Renvoie la première colonne de x
    x2 = x(:,end);
end
```

La fonction Channel2 est un peu particulière. En soit, elle ne renvoit pas vraiment la deuxième piste audio du signal, mais la dernière colonne de ce dernier. Ainsi, si le signal est mono, le résultat de Channel2 sera identique à Channel1. (rmq: à noter que Channel2 renvoie bien la dernière colonne. Un signal de type multi-piste supérieur à deux ne sera pas compatible avec le programme. Seul les musiques mono et stéréo le sont.)

* Dualchannel

```
function i = dualchannel(x)
% fonction dualchannel renvoie un boolean 0 ou 1 :
% si le signal x est de type mono, i = 1
% si le signal x est de type stéréo, i = 2
x1 = channel1(x);
x2 = channel2(x);

if(x1 == x2)
    i=1;
else
    i=2;
end
```

Dualchannel vérifie le commentaire fait précédement : Si le résultat de Channel1 est identique à celui de Channel2, le signal est mono, on renvoie 1. Si les résultats sont différents, le signal est stéréo, on renvoie 2.

* stereotomono

```
function x = stereotomono(x)
% fonction stereo to mono transforme un son stereo en mono
% le signal x devient la résultante des deux channels 1 et 2
% représentant la partie gauche (x1) et droite (x2) du signal
x1 = channel1(x);
x2 = channel2(x);

x = x1 + x2;
```

stereotomono est additione les deux matrices channel1 et channel2 afin de les obtenir sur une seul colonne. Le resultat transforme le signal stereo en un signal mono.

* getvariable

```
% Si la musique selectionné est 2
if(handles.selectmusique == 2)
    % x et Fs seront ce de obj2
    x = handles.obj2x;
    Fs = handles.obj2fe;
else
    % sinon, x et Fs seront ce de obj1
    x = handles.obj1x;
    Fs = handles.obj1fe;
end
```

Le script getvariable est utilisé à chaque début d'effet. En fonction de la musique demandé pour l'effet, les variables x et Fs seront enregistré par rapport à la musique 1 ou 2.

* savedata

```
if(handles.selectmusique == 2)
    handles.obj2aff = y;
    handles.obj2tmp = 0:1/Fs:(length(y)-1)/Fs;
    handles.obj2audio = audioplayer((y),Fs);
    handles.obj2fe = Fs;
    guidata(hObject,handles);
else
    handles.obj1aff = y;
    handles.obj1tmp = 0:1/Fs:(length(y)-1)/Fs;
    handles.obj1audio = audioplayer(y,Fs);
    handles.obj1fe = Fs;
    guidata(hObject,handles);
end
```

Le script savedata est utilisé à chaque fin d'effet. En fonction de la musique demandé pour l'effet, les résultats produit par l'effet sont sauvegardé dans l'objet musique correspondant.

3) Les effets

Maintenant que le fonctionnement du programme est mis en place, les effets doivent être implémenté par catégorie : Temporel, Fréquentielle, et Dynamique.

A. Les effets Temporel

● Effet Écho :

```
getvariable;

% temps de retard de l'echo : Fs*0,4 => 2/5 de la fréquence
t = round(Fs*0.4);

% à partir du retard, jusqu'à la fin du signal, on sauvegarde le décalage à amplitude 0,3
y(:,1) = zeros(size(x(:,1)));
for i = (t+1):length(x(:,1)),
      y(i,1) = 0.3*x(i-t,1);
end

% STEREO
if(dualchannel(x) == 2)
   y(:,2) = zeros(size(x(:,2)));
   for i = (t+1):length(x(:,2)),
         y(i,2) = 0.3*x(i-t,2);
   end
end

% résultat final signal retardé + signal original
y = y + x;

% sauvegarde

savedata;
```

Pour créer l'effet écho, on additionne le signal d'origine avec ce même signal mais atténué et retardé. Le resultat final donne une impression de persistance du son en fond, donnant l'effet d'écho.

●Effet Flanger

```
getvariable;
% taille du signal
taille = 1:length(x(:,1));

% sinusoidale modulante
modul = sin(2*pi*taille*(1/Fs));

% temps de retard -> 0,0008*Fs = 1/1250 de la fréquence soit extrèmement court
t = round(0.0008*Fs);

% Initialisation du signal final
y(:,1) = zeros(size(x(:,1)));

% jusqu'au retard : y(signal final) = x(signal audio)
y(1:t,1)=x(1:t,1);

% à partir du retard, jusqu'à la fin du signal, on applique la modulation
for i = (t+1):length(x(:,1)),
    %valeur absolue entière du signal modulant multiplié par le retard
    abs_modul_retard=round(abs(modul(i))*t);
    % signal final = signal audio original + signal retardé par rapport au retard temporel modulé
    y(i,1) = x(i,1) + 0.5*(x(i-abs_modul_retard,1));
end
% STEREO
if(dualchannel(x) == 2)
    % taille du signal
    taille = 1:length(x(:,2));
    % sinusoidale modulante
    modul2 = sin(2*pi*taille*(1/Fs));
    % jusqu'au delai : y(signal final) = x(signal audio)
    y(1:t,2)=x(1:t,2);
    % à partir du delai, jusqu'à la fin du signal, on applique la modulation
    for i = (t+1):length(x(:,2)),
        %valeur absolue entière du signal modulant multiplié par le retard
        abs_modul_retard=round(abs(modul2(i))*t);
        % signal final = signal audio original + signal retardé par rapport au retard temporel modulé
        y(i,2) = x(i,2) + 0.5*(x((i-abs_modul_retard),2));
    end
end

% sauvegarde

savedata;
```

Pour créer l'effet flanger, on additionne le signal d'origine avec ce même signal mais légèrement retardé. Le retard agis de manière particulière ici : multiplié par une sinusoide, ce dernier varie périodiquement. Donnant un retard variant comme un "accordéon" (de plus en plus élevé, pour redevenir de plus en plus faible...etc)

B. Les effets Fréquentielle

● Effet Wah-wah

```
getvariable;

% frequence min et max de l'effet wah wah
fmin=400;
fmax=4000;

% frequence de base de l'effet wah wah : (fmin+fmax)/2
Fw = 2200;

%delta fréquence entre base de l'effet et celle du signal audio
delta = Fw/Fs;

% Tableau de fréquence allant de fmin à fmax avec pour pas de variation delta
Fc=fmin:delta:fmax;

% Tant que la taille de Fc est inférieur à celle de x, Fc = variation de Fmax -> fmin et fmin -> fmax par
% pas de delta
while(length(Fc) < length(x) )
    Fc= [ Fc (fmax:-delta:fmin) ];
    Fc= [ Fc (fmin:delta:fmax) ];
end

% Taille de Fc = les valeurs de Fc(1) jusqu'à Fc(taille de x)
Fc = Fc(1:length(x(:,1)));
% Coeff Frequence : différent pour chaque valeur de Fc
F1 = 2*sin((pi*Fc(1))/Fs);
% Coeff d'amplitude du filtre
Q1 = 2*0.05;
%vecteur resultat (3 pour correspondre à la fréquence trianglue du wah wah)
yh=zeros(size(x(:,1)));
yb(:,1)=zeros(size(x(:,1)));
yl=zeros(size(x(:,1)));
% Première valeur pour ne pas être nul à cause du retard n-1
yh(1) = x(1,1);
yb(1,1) = F1*yh(1);
yl(1) = F1*yb(1,1);

% Calcul
for n=2:length(x(:,1)),
    yh(n) = x(n,1) - yl(n-1) - Q1*yb(n-1,1);
    yb(n,1) = F1*yh(n) + yb(n-1,1);
    yl(n) = F1*yb(n,1) + yl(n-1);
    F1 = 2*sin((pi*Fc(n))/Fs);
end
```

```
% STEREO
if(dualchannel(x) == 2)
   Fc = Fc(1:length(x(:,2)));
   F1 = 2*sin((pi*Fc(1))/Fs);
   yh=zeros(size(x(:,2)));
   yb(:,2)=zeros(size(x(:,2)));
   yl=zeros(size(x(:,2)));

   yh(1) = x(1,2);
   yb(1,2) = F1*yh(1);
   yl(1) = F1*yb(1,2);

   for n=2:length(x(:,2)),
      yh(n) = x(n,2) - yl(n-1) - Q1*yb(n-1,2);
      yb(n,2) = F1*yh(n) + yb(n-1,2);
      yl(n) = F1*yb(n,2) + yl(n-1);
      F1 = 2*sin((pi*Fc(n))/Fs);
   end
maxyb(:,2) = max(abs(yb(:,2)));
y(:,2) = yb(:,2)/maxyb(:,2);
end

maxyb(:,1) = max(abs(yb(:,1)));
y(:,1) = yb(:,1)/maxyb(:,1);

savedata;
```

L'effet Wah wah agi de manière très particulière sur le signal. Celui ci est modifié en fréquence de manière perpetuel. Mais sur des valeurs délimité entre fmin et fmax par un pas delta étant la variation de ces fréquence par rapport à celle du signal. Ces changement de fréquence sont eux même modulé par une sinusoide qui va ainsi modifié la puissance de variation, on ce retrouve donc avec des morceaux variant de manière périodique.

Pour l'exemple d'une guitare, la variation de Fréquence Fc va jouer de rôle de l'effet, et le fait de l'encapsuler dans une sinusoïde permet de remplacer la pédale, et ainsi faire varier les fréquences vers de manière forte ou non, comme si l'on jouait avec la pédale.

Le calcul final y = yb/maxyb est du au fait que yb par sa variation en continu, sort du champ de résultat de y. Afin de garder le son dans un état écoutable, ce dernier est divisé par la valeur maximum pour ne pas recontrer de problème.

●Effet de Vitesse

```
getvariable;
if(handles.selectmusique == 1)
% Si le bouton de vitesse de la musique 1 est sélectionné, on modifie la valeur de la fréquence
   if(handles.obj1speed == 1)
      varspeed = handles.obj1acc;
   else
      varspeed = handles.obj1dec;
   end
   Fs = handles.obj1fo*varspeed;
else
% Si le bouton de vitesse de la musique 2 est sélectionné, on modifie la valeur de la fréquence
   if(handles.obj2speed == 1)
      varspeed = handles.obj2acc;
   else
      varspeed = handles.obj2dec;
   end
   Fs = handles.obj2fo*varspeed;
end
y = x;
% sauvegarde

savedata;
```

Pour obtenir l'effet d'accélération ou de décélération, on augmente ou diminue la fréquence du signal d'entré selon le cas.

Dans le cas de l'accélération, par pas de 0,1 : La fréquence est multiplié par un coéfficient variant entre 1 et 2, rendant la musique jusqu'à deux fois plus vite.

Dans le cas de la décélération, par pas de 0,1 : La fréquence est multiplié par un coéfficient variant entre 1 et 0,5 rendant la musique jusqu'à deux fois plus lente

C. Les effets Dynamique

●Effet de séparation des écouteurs

```
x2 = handles.obj2aff;
if(dualchannel(x2) == 2)
   x2 = stereotomono(x2);
end
x2z = zeros(size(x2));
Fs2 = handles.obj2fe;

x1 = handles.obj1aff;
if(dualchannel(x1) == 2)
   x1 = stereotomono(x1);
end
x1z= zeros(size(x1));
Fs1 = handles.obj1fe;

if(handles.separatemode == 0)

handles.obj1audio = audioplayer([x1,x1z], Fs1);
handles.obj2audio = audioplayer([x2z,x2], Fs2);
handles.separatemode = 1;
guidata(hObject,handles);
stop(handles.obj1audio);
stop(handles.obj2audio);

else

handles.obj1audio = audioplayer(x1, Fs1);
handles.obj2audio = audioplayer(x2, Fs2);
stop(handles.obj1audio);
stop(handles.obj2audio);
handles.separatemode = 0;
guidata(hObject,handles);
end
```

Dans le cas d'une musique en stéréo, nous nous sommes heurté à plusieurs problèmes. Mais après compréhension total du fonctionnement d'un signal en stéréo, l'idée de mettre en place un mode de séparation nous est apparu.
Le but ici est de convertir les deux sons choisi en mono, et les mettre en place dans la colonne 1 pour l'écouteur gauche, et la colonne 2 pour l'écouteur droit.

Néanmoins, mettre les deux signaux dans un seul objet audioplayer ne fonctionnait pas pour une raison inconnu. Pour palier à ce problème, mettre en place deux objets audioplayer a résolu le problème.

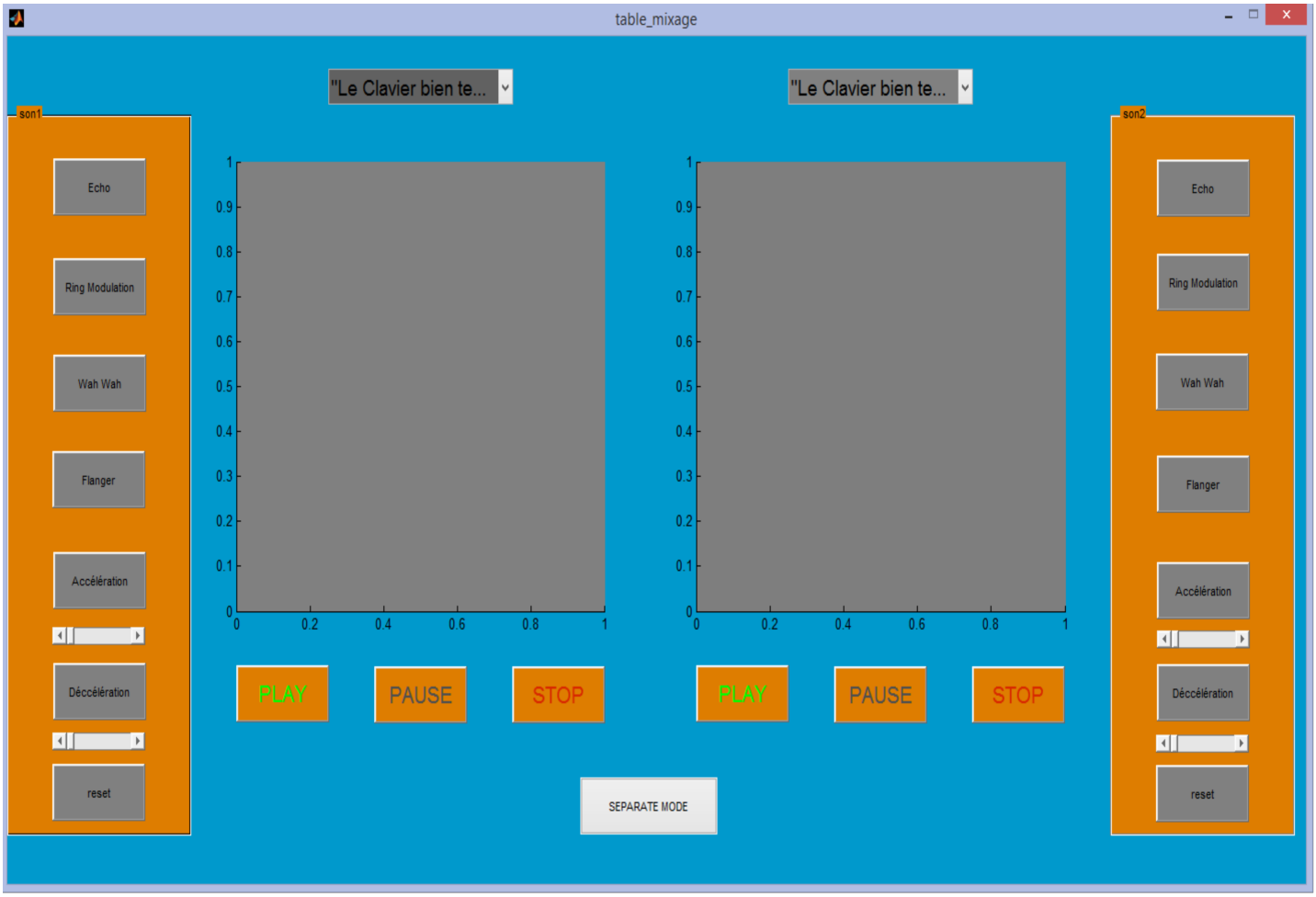
●Effet Ringmode

```
getvariable;
% taille du signal audio x
taille = 1:length(x(:,1));

% Ring Modulator avec fréquence 880Hz
Fc = 880;
mod(:,1) = sin(2*pi*taille*(Fc/Fs));

% faire Ring Modulation
y(:,1) = x(:,1) .* mod;

% STEREO
if(dualchannel(x) == 2)

   taille = 1:length(x(:,2));
   carrier(:,1) = sin(2*pi*taille*(Fc/Fs));
   y(:,2) = x(:,2) .* mod;

end
% sauvegarde
savedata;
```

L'effet Ring Modulator permet d'obtenir un son robotique en multipliant les valeurs du signal audio par une sinusoïde ayant un rapport Fc/Fs avec Fs une valeur fixe de 880Hz et Fs la fréquence du signal.

Ce rapport donne une valeur à unité ni fréquentielle, ni temporel, ce qui permet à la multiplication avec le signal de donner un son unique travaillant non pas sur la fréquence ou le temps, mais plus la note du signal original. La sinusoïde permettant de faire des variations périodique cela donne un rendu plutot original.

III. L'affichage



Notre interface a été créé de tel sorte que l'on puisse jouer deux sons simultanément, tout en ayant la possibilité d'effectuer les effets sur les sons proposés.

De manière intuitive, chaque sons à son propre tableau de commande de son coté, ainsi que des boutons de contrôle en dessous du graphique représentant le signal par rapport aux temps.

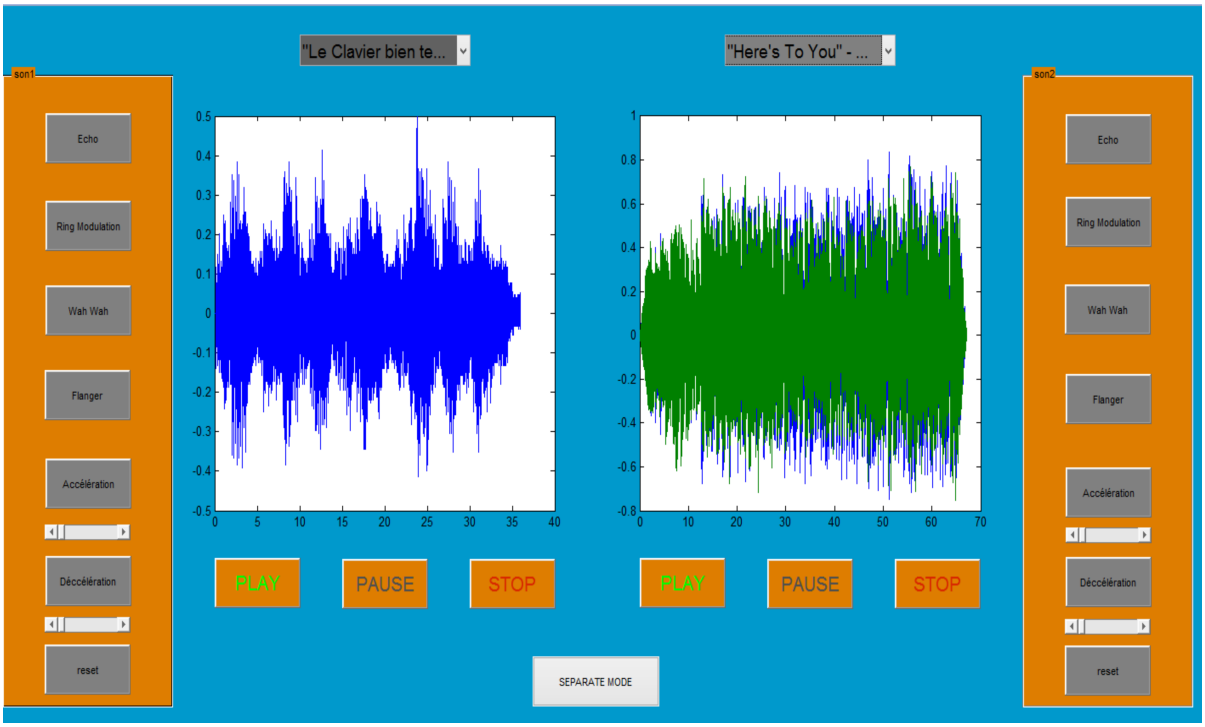
Au démarrage, il faut choisir une musique parmis les menus déroulants afin de faire apparaitre son graphique. Une fois ce dernier apparu, il suffit de cliquer sur un bouttons pour actionner son fonctionnement, et enfin, appuyer sur "PLAY" pour lancer la musique.

Utilisation :

- Les deux "list-Box" permettent de choisir le son voulu.

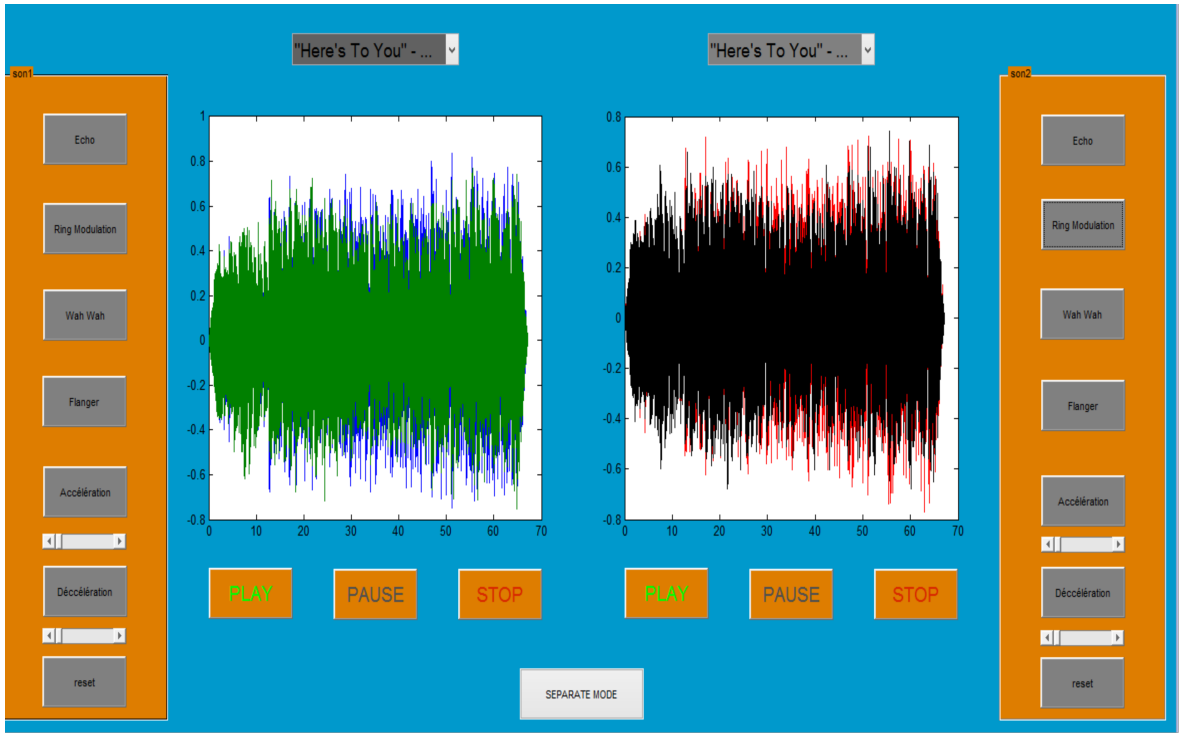
Gérer par un switch – case, la musique choisi se charge en donné pour être ensuite manipulé. Exemple :

```
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
val = get(handles.popupmenu1,'value');
% determine une valeur dans le popmenu associÃ©
switch(val)
case 1
[x2,Fe2,Nbits2]=wavread('C:\projet_elec\piano');
% ici on charge le son
handles.obj2x=x2;
handles.obj2fo=Fe2;
handles.obj2fe=Fe2;
handles.obj2n=Nbits2;
handles.obj2audio = audioplayer(x2,Fe2);
handles.obj2tmp=0:1/Fe2:(length(x2)-1)/Fe2;
handles.obj2aff=x2;
guidata(hObject,handles);
axes(handles.axes2);
plot(handles.obj2tmp,x2);
```



- Sur les colonnes se trouve les différents effets proposés.
  En appuyant sur le bouttons d'un effet, ce dernier se charge en donnée grace aux scripts, mais la musique ne se lance pas.
  Exemple :

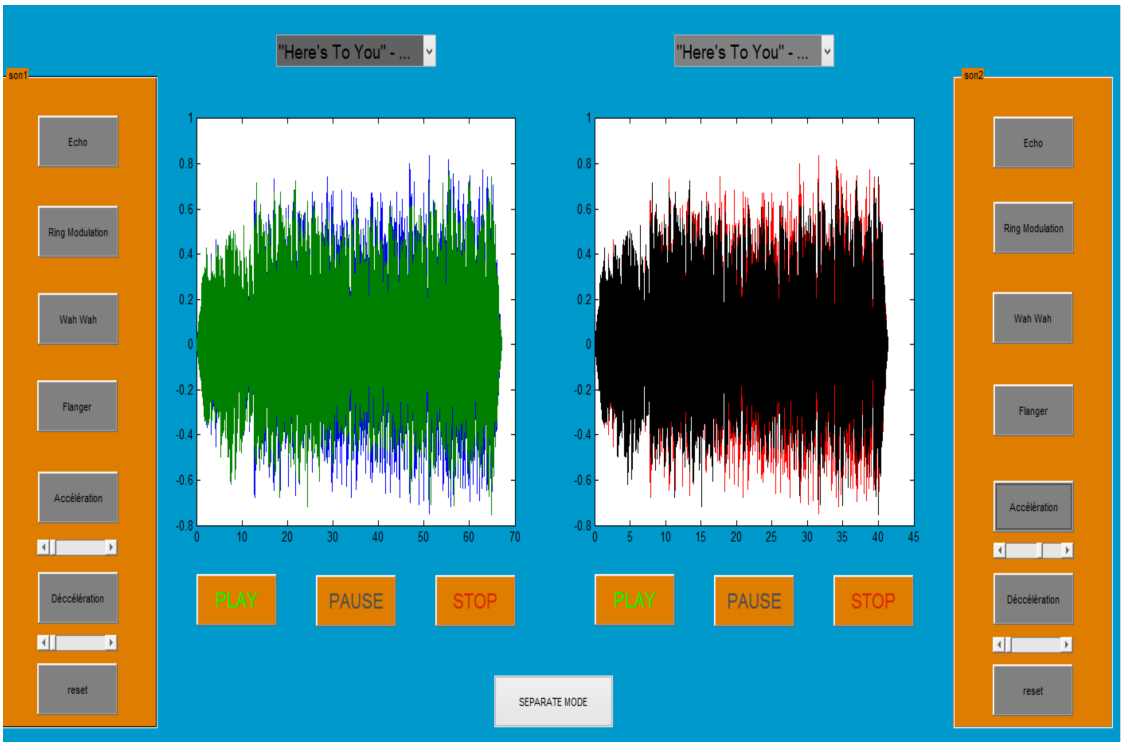
```
% --- Executes on button press in pushbutton12. / RINGMOD 2
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.selectmusique = 2;
guidata(hObject,handles);
ringmod
axes(handles.axes2);
if(dualchannel(handles.obj2aff)==2)
    plot(handles.obj2tmp,handles.obj2aff(:,1),'r',
    handles.obj2tmp,handles.obj2aff(:,2),'k');
else
    plot(handles.obj2tmp,handles.obj2aff,'r');
end
```



- Dans les colonnes des effets se trouve deux slider.

Ces derniers permettent de gérer l'accélération ou décélération
Exemple :

```
% --- Executes on button press in pushbutton14. / ACCELERATION 2
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.selectmusique = 2;
handles.obj2speed = 1;
guidata(hObject,handles);
speed
axes(handles.axes2);
if(dualchannel(handles.obj2aff)==2)
    plot(handles.obj2tmp,handles.obj2aff(:,1),'r',
    handles.obj2tmp,handles.obj2aff(:,2),'k');
else
    plot(handles.obj2tmp,handles.obj2aff,'r');
end

    % --- Executes on slider movement.
    function slider10_Callback(hObject, eventdata, handles)
    % hObject    handle to slider10 (see GCBO)
    % eventdata  reserved - to be defined in a future version of MATLAB
    % handles    structure with handles and user data (see GUIDATA)
    handles.obj2acc = 1 + get(hObject,'Value');
    guidata(hObject,handles);
```



- Sous les graphiques se trouve les bouttons de gestion du sons

Les bouttons PLAY, PAUSE, STOP permettant respectivement de lancer la musique, la mettre en pause, et l'arrêter. (Rmq : PLAY à la particularité de reprendre la musique si celle si était sur pause avant.) Exemple :

```
% --- Executes on button press in pushbutton20. / PLAY-RESUME 2
function pushbutton20_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if(handles.obj2pauseid == 1)
    resume(handles.obj2audio);
    handles.obj2pauseid = 2;
    guidata(hObject,handles);
else
    play(handles.obj2audio);
    handles.obj2pauseid = 2;
    guidata(hObject,handles);
end
```

- Tout en bas du programme se trouve le boutton separate mode

ce dernier permet jouer chacune des musiques dans un écouteur différent. En revanche, matlab execute les instructions ligne par ligne, et lorsqu'une musiques est trop volumineuse, il y a un décalage au démarrage des deux musiques, pour palier au problème , nous avons fait une manipulation un peu bricolage :

```
% --- Executes on button press in pushbutton23. / SEPARATE MODE
function pushbutton23_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
    separate
    if(handles.separatemode == 1)
        play(handles.obj1audio);
        pause(handles.obj1audio);
        play(handles.obj2audio);
        resume(handles.obj1audio);
    end
```

IV. Conclusion

Le Cloirec Youen :

Personnellement, ce projet sur le traitement de signal m'a permis de faire connaissance avec le logiciel Matlab, entrevoir ses capacités et nottament dans la création d'effets sonores.
Cela m'a également permis de mettre en application les théories du cours et de voir l'aspect concret du traitement de signal.
Pour ce qui est du travail, je me suis très bien adapté avec la façon de travailler de mes camarades et le partage des tâches à effectuer.
J'ai pris un grand plaisir à réaliser ce projet avec Léopold et Vincent.

Grosjean Léopold :

J'ai pris un grand plaisir à réaliser ce projet avec mes camarades de classe que j'ai rencontré cette année. De plus, le sujet était intéressant et m'a ainsi permis de garder ma motivation tout au long de la réalisation de notre projet. Ayant de nombreuses idées d'option à ajouter au projet, nous avons d'un commun accord décidé de réaliser une « table de mixage » permettant de modifier deux musiques simultanément et possédant une interface très facile à utiliser.
Nous nous sommes tout de suite très bien entendu et je n'ai pas, ou peu rencontré de grosses difficultés vis à vis du logiciel Matlab. En effet, Vincent venant d'un IUT d'informatique, a tout de suite su maitriser le logiciel que j'avais déjà manipulé grâce à mon électif «Traitement de l'image» et pu m'aider dès que je rencontrais un problème.
Cependant, la compréhension de l'algorithme du « Wah-wah » fût laborieuse pour ma part. J'aspire à retravailler avec ce même groupe sur d'autres projets, notre leitmotiv est « Le travail paye ! ».

Kocupyr Vincent :

Ayant déjà travaillé avec Léopold sur d'autre projet, nous avons tout de suite trouver une ambiance de travail très convenable. Utiliser matlab pour la première fois est un peu déstabilisant mais le challenge proposé par le sujet et la liberté sur celui ci m'a permis d'exploiter mes connaissances en programmation pour faire de ce projet plus un jeux qu'un travail théorique. Nous nous sommes imposé certaines chose, quitte à en laisser d'autre de coté, mais au final je trouve le résultat satisfaisant.
Mon regret serait de n'avoir pas eu plus de temps pour mettre d'autres d'effets. La maitrise de matlab et l'interface GUI à pris plus de temps que je ne l'aurai cru, et maitre tout en place fut le plus long.
En revanche, l'avantage de notre projet est d'avoir cette aptitude à s'adapter. Ayant une base solide. Rajouter des effets demande d'utiliser les deux scripts générique getvariable en début, et savedata à la fin, et mettre le code de l'effet entre.
Le projet étant terminé, je pense continuer à travailler dessus encore quelques jours voir jusqu'ou est il possible de le pousser.

Source :

http://idf.udppc.asso.fr/IMG/pdf/WahWah1_2_3_4.pdf
http://users.rowan.edu/~shreek/networks1/music.html
http://www.cs.cf.ac.uk/
http://www.mathworks.fr/fr/help/matlab/
http://www.wikipedia.com