



Accueil > Cours > Apprenez à programmer en C ! > TP : visualisation spectrale du son

# Apprenez à programmer en C !

40 heures Moyenne

Mis à jour le 25/04/2019



## TP : visualisation spectrale du son

Ce chapitre de travaux pratiques va vous proposer de manipuler la SDL et FMOD simultanément. Cette fois, nous n'allons pas travailler sur un jeu. Certes, la SDL est tout particulièrement adaptée à cela, mais on peut l'utiliser dans d'autres domaines. Ce chapitre va justement vous prouver qu'elle peut servir à autre chose.

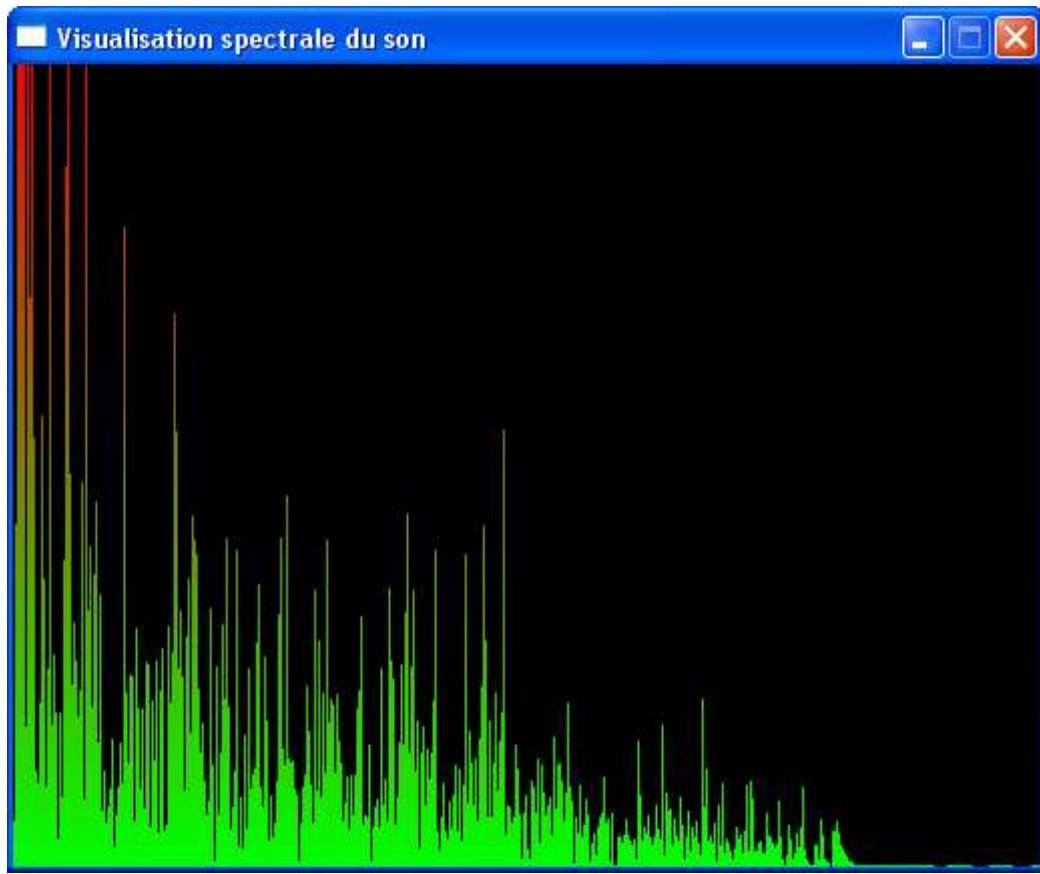
Nous allons réaliser ici une visualisation du spectre sonore en SDL. Cela consiste à afficher la composition du son que l'on joue, par exemple une musique. On retrouve cela dans de nombreux lecteurs audio. C'est amusant et ce n'est pas si compliqué que ça en a l'air !

Ce chapitre va nous permettre de travailler autour de notions que nous avons découvertes récemment :

- la gestion du temps ;
- la bibliothèque FMOD.

Nous découvrirons en outre comment modifier une surface pixel par pixel.

La figure suivante vous donne un aperçu du programme que nous allons créer dans ce chapitre.



C'est le genre de visualisation qu'on peut retrouver dans des lecteurs audio tels que Winamp, Windows Media Player ou encore Amarok.

Et pour ne rien gâcher, comme je vous l'ai dit ce n'est pas bien difficile à faire. D'ailleurs, contrairement au TP Mario Sokoban, cette fois c'est vous qui allez travailler. Ça vous fera un très bon exercice.

## Les consignes

Les consignes sont simples. Suivez-les pas à pas dans l'ordre, et vous n'aurez pas d'ennuis.

### 1/ Lire un MP3

Pour commencer, vous devez créer un programme qui lit un fichier MP3. Vous n'avez qu'à reprendre [la chanson « Home » du groupe Hype](#) que nous avons utilisée dans le chapitre sur FMOD pour illustrer le fonctionnement de la lecture d'une musique.

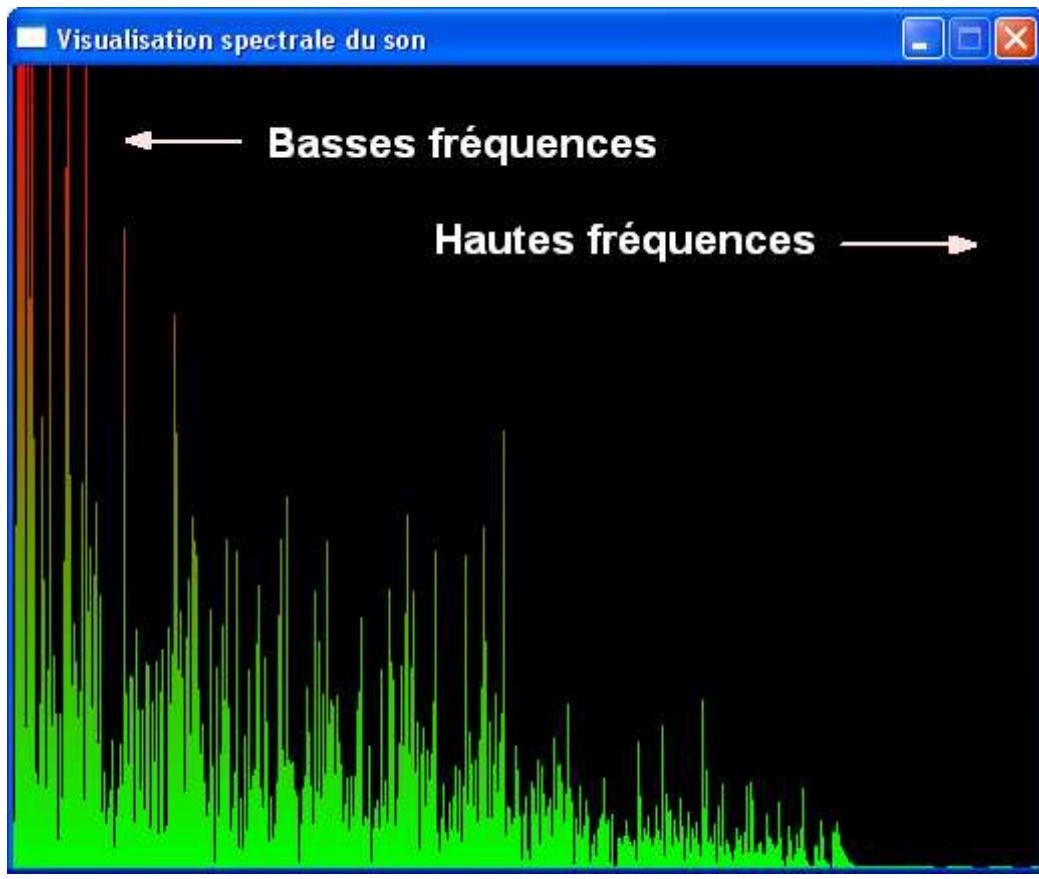
Si vous avez bien suivi le chapitre sur FMOD, il ne vous faudra pas plus de quelques minutes pour arriver à le faire. Je vous conseille au passage de placer le MP3 dans le dossier de votre projet.

### 2/ Récupérer les données spectrales du son

Pour comprendre comment la visualisation spectrale fonctionne, il est indispensable que je vous explique un peu comment cela fonctionne à l'intérieur (pas dans le détail non plus, sinon ça va se transformer en cours de maths).

Un son peut être découpé en fréquences. Certaines fréquences sont basses, d'autres moyennes, et d'autres hautes.

Ce que nous allons faire dans notre visualisation, c'est afficher la quantité de chacune de ces fréquences sous forme de barres. Plus la barre est grande, plus la fréquence est utilisée (figure suivante).



Sur la gauche de la fenêtre, nous faisons donc apparaître les basses fréquences, et sur la droite les hautes fréquences.

Mais comment récupérer les quantités de chaque fréquence ?

FMOD nous mâche le travail. On peut faire appel à la fonction `FMOD_Channel_GetSpectrum`. Son prototype est le suivant :

```
1 FMOD_RESULT FMOD_Channel_GetSpectrum(  
2     FMOD_CHANNEL * channel,  
3     float * spectrumarray,  
4     int numvalues,  
5     int channeloffset,  
6     FMOD_DSP_FFT_WINDOW windowtype  
7 );
```

Et voici ses paramètres :

- Le canal sur lequel la musique est jouée. Donc *a priori* il faut récupérer un pointeur vers ce canal.
- Un tableau de `float`. Il faut que ce tableau soit déjà alloué, statiquement ou dynamiquement, pour permettre à FMOD de le remplir correctement.

- La taille du tableau. Cette taille doit obligatoirement être une puissance de 2, par exemple 512.
- Ce paramètre sert à définir à quelle sortie on s'intéresse. Par exemple si vous êtes en stéréo, 0 veut dire gauche, et 1 veut dire droite.
- Ce paramètre est un peu plus complexe, et ne nous intéresse pas vraiment dans ce cours. On se contentera de lui donner la valeur `FMOD_DSP_FFT_WINDOW_RECT`.

Rappel : le type `float` est un type décimal, au même titre que `double`. La différence entre les deux vient du fait que `double` est plus précis que `float`, mais dans notre cas, le type `float` est suffisant. C'est celui utilisé par FMOD ici, donc c'est celui que nous devrons utiliser nous aussi.

En clair, on déclare notre tableau de `float` :

```
1 float spectre[512];
```

Ensuite, lorsque la musique est en train d'être jouée, on demande à FMOD de remplir le tableau du spectre en faisant par exemple :

```
1 FMOD_Channel_GetSpectrum(canal, spectre, 512, 0, FMOD_DSP_FFT_WINDOW_RECT);
```

On peut ensuite parcourir ce tableau pour obtenir les valeurs de chacune des fréquences :

```
1 spectre[0] // Fréquence la plus basse (à gauche)
2 spectre[1]
3 spectre[2]
4 ...
5 spectre[509]
6 spectre[510]
7 spectre[511] // Fréquence la plus haute (à droite)
```

Chaque fréquence est un nombre décimal compris entre 0 (rien) et 1 (maximum). Votre travail va consister à afficher une barre plus ou moins grande en fonction de la valeur que contient chaque case du tableau.

Par exemple, si la valeur est 0.5, vous devrez tracer une barre dont la hauteur correspondra à la moitié de la fenêtre.

Si la valeur est 1, elle devra faire toute la hauteur de la fenêtre.

Généralement, les valeurs sont assez faibles (plutôt proches de 0 que de 1). Je recommande de multiplier par 20 toutes les valeurs pour mieux voir le spectre.

Attention : si vous faites ça, vérifiez que vous ne dépassez pas 1 (arrondissez à 1 s'il le faut). Si vous vous retrouvez avec des valeurs supérieures à 1, vous risquez d'avoir des problèmes pour tracer les barres verticales par la suite !

Mais les barres doivent bouger au fur et à mesure du temps non ? Comme le son change tout le temps, il faut mettre à jour le graphique. Comment faire ?

Bonne question. En effet, le tableau de 512 `float` que vous renvoie FMOD **change toutes les 25 ms** (pour être à jour par rapport au son actuel). Il va donc falloir dans votre code que vous relisez le tableau de 512 floats (en refaisant appel à `FMOD_Channel_GetSpectrum` toutes les 25 ms), puis que vous mettiez à jour votre graphique en barres.

Relisez le chapitre sur la gestion du temps en SDL pour vous rappeler comment faire. Vous avez le choix entre une solution à base de `GetTicks` ou à base de callbacks. Faites ce qui vous paraît le plus facile.

## 4/ Réaliser le dégradé

Dans un premier temps, vous pouvez réaliser des barres de couleur unie. Vous pourrez donc créer des surfaces. Il devra y avoir 512 surfaces : une pour chaque barre. Chaque surface fera donc 1 pixel de large et la hauteur des surfaces variera en fonction de l'intensité de chaque fréquence.

Toutefois, je vous propose ensuite d'effectuer une amélioration : la barre doit tendre vers le rouge lorsque le son devient de plus en plus intense. En clair, la barre doit être verte en bas et rouge en haut.

Mais... une surface ne peut avoir qu'une seule couleur si on utilise `SDL_FillRect()`. On ne peut pas faire de dégradé !

En effet. On pourrait certes créer des surfaces de 1 pixel de large et 1 pixel de haut pour chaque couleur du dégradé, mais ça ferait vraiment beaucoup de surfaces à gérer et ce ne serait pas très optimisé !

Comment fait-on pour dessiner pixel par pixel ?

Je ne vous l'ai pas appris auparavant, car cette technique ne méritait pas un chapitre entier. Vous allez voir en effet que ce n'est pas bien compliqué.

En fait, la SDL ne propose aucune fonction pour dessiner pixel par pixel. Mais on a le droit de l'écrire nous-mêmes.

Pour ce faire, il faut suivre ces étapes méthodiquement dans l'ordre :

1. Faites appel à la fonction `SDL_LockSurface` pour annoncer à la SDL que vous allez modifier la surface manuellement. Cela « bloque » la surface pour la SDL et vous êtes le seul à y avoir accès tant que la surface est bloquée.

Ici, je vous conseille de ne travailler qu'avec une seule surface : l'écran. Si vous voulez dessiner un pixel à un endroit précis de l'écran, vous devrez donc bloquer la surface `ecran` :

```
1 SDL_LockSurface(ecran);
```

1. 1. Vous pouvez ensuite modifier le contenu de chaque pixel de la surface. Comme la SDL ne propose aucune fonction pour faire ça, il va falloir l'écrire nous-même dans notre programme.

Cette fonction, je vous la donne. Je la tire de la documentation de la SDL. Elle est un peu compliquée car elle travaille sur la surface directement et gère toutes les profondeurs de couleurs (bits par pixel) possibles. Pas besoin de la retenir ou de la comprendre, copiez-la simplement dans votre programme pour pouvoir l'utiliser :

```
1 void setPixel(SDL_Surface *surface, int x, int y, Uint32 pixel)
2 {
3     int bpp = surface->format->BytesPerPixel;
4
5     Uint8 *p = (Uint8 *)surface->pixels + y * surface->pitch + x * bpp;
6
7     switch(bpp) {
8         case 1:
9             *p = pixel;
10            break;
11
12        case 2:
13            *(Uint16 *)p = pixel;
14            break;
15
16        case 3:
17            if(SDL_BYTEORDER == SDL_BIG_ENDIAN) {
18                p[0] = (pixel >> 16) & 0xff;
19                p[1] = (pixel >> 8) & 0xff;
20                p[2] = pixel & 0xff;
21            } else {
22                p[0] = pixel & 0xff;
23                p[1] = (pixel >> 8) & 0xff;
24                p[2] = (pixel >> 16) & 0xff;
25            }
26            break;
27
28        case 4:
29            *(Uint32 *)p = pixel;
30            break;
31    }
32 }
```

Elle est simple à utiliser. Envoyez les paramètres suivants :

- le pointeur vers la surface à modifier (cette surface doit préalablement avoir été bloquée avec `SDL_LockSurface`);
- la position en abscisse du pixel à modifier dans la surface (x) ;
- la position en ordonnée du pixel à modifier dans la surface (y) ;

- la nouvelle couleur à donner à ce pixel. Cette couleur doit être au format `Uint32` ; vous pouvez donc la générer à l'aide de la fonction `SDL_MapRGB()` que vous connaissez bien maintenant.
- Enfin, lorsque vous avez fini de travailler sur la surface, il ne faut pas oublier de la débloquer en appelant `SDL_UnlockSurface`.

1 `SDL_UnlockSurface(ecran);`

## Code résumé d'exemple

Si on résume, vous allez voir que c'est tout simple.

Ce code dessine un pixel rouge au milieu de la surface `ecran` (donc au milieu de la fenêtre).

```
1 SDL_LockSurface(ecran); /* On bloque la surface */
2 setPixel(ecran, ecran->w / 2, ecran->h / 2, SDL_MapRGB(ecran->format, 255, 0, 0)); /* On dessine un
pixel rouge au milieu de l'écran */
3 SDL_UnlockSurface(ecran); /* On débloque la surface*/
```

Avec cette base vous devriez pouvoir réaliser des dégradés du vert au rouge (un indice : il faut utiliser des boucles 🎉).

## La solution

Alors, comment avez-vous trouvé le sujet ? Il n'est pas bien difficile à appréhender, il faut juste faire quelques calculs, surtout pour la réalisation du dégradé. C'est du niveau de tout le monde, il faut juste réfléchir un petit peu.

Certains mettent plus de temps que d'autres pour trouver la solution. Si vous avez du mal, ce n'est pas bien grave. Ce qui compte, c'est de finir par y arriver. Quel que soit le projet dans lequel vous vous lancerez, vous aurez forcément des moments où il ne suffit pas de savoir programmer ; il faut aussi être logique et bien réfléchir.

Je vous donne le code complet ci-dessous. Il est suffisamment commenté.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <SDL/SDL.h>
4 #include <fmodex/fmod.h>
5
6 #define LARGEUR_FENETRE      512 /* DOIT rester à 512 impérativement car il y a 512 barres
(correspondant aux 512 floats) */
7 #define HAUTEUR_FENETRE      400 /* Vous pouvez la faire varier celle-là par contre */
8 #define RATIO                (HAUTEUR_FENETRE / 255.0)
9 #define DELAI_RAFRAICHISSEMENT 25 /* Temps en ms entre chaque mise à jour du graphe. 25 ms est la
valeur minimale. */
10 #define TAILLE_SPECTRE       512
11
12 void setPixel(SDL_Surface *surface, int x, int y, Uint32 pixel);
13
14 int main(int argc, char *argv[])
```

```
15 {
16     SDL_Surface *ecran = NULL;
17     SDL_Event event;
18     int continuer = 1, hauteurBarre = 0, tempsActuel = 0, tempsPrecedent = 0, i = 0, j = 0;
19     float spectre[TAILLE_SPECTRE];
20
21     /* Initialisation de FMOD
22     -----
23
24     On charge FMOD, la musique on lance la lecture de la musique.
25
26 */
27
28     FMOD_SYSTEM *system;
29     FMOD_SOUND *musique;
30     FMOD_CHANNEL *canal;
31     FMOD_RESULT resultat;
32
33
34     FMOD_System_Create(&system);
35     FMOD_System_Init(system, 1, FMOD_INIT_NORMAL, NULL);
36
37     /* On ouvre la musique */
38     resultat = FMOD_System_CreateSound(system, "hype_home.mp3", FMOD_SOFTWARE | FMOD_2D |
39     FMOD_CREATESTREAM, 0, &musique);
40
41     /* On vérifie si elle a bien été ouverte (IMPORTANT) */
42     if (resultat != FMOD_OK)
43     {
44         fprintf(stderr, "Impossible de lire le fichier mp3\n");
45         exit(EXIT_FAILURE);
46     }
47
48     /* On joue la musique */
49     FMOD_System_PlaySound(system, FMOD_CHANNEL_FREE, musique, 0, NULL);
50
51     /* On récupère le pointeur du canal */
52     FMOD_System_GetChannel(system, 0, &canal);
53
54     /* Initialisation de la SDL
55     -----
56
57     On charge la SDL, on ouvre la fenêtre et on écrit dans sa barre de titre
58     On récupère au passage un pointeur vers la surface ecran
59     Qui sera la seule surface utilisée dans ce programme */
60
61     SDL_Init(SDL_INIT_VIDEO);
62     ecran = SDL_SetVideoMode(LARGEUR_FENETRE, HAUTEUR_FENETRE, 32, SDL_SWSURFACE | SDL_DOUBLEBUF);
63     SDL_WM_SetCaption("Visualisation spectrale du son", NULL);
64
65     /* Boucle principale */
66
67     while (continuer)
68     {
69         SDL_PollEvent(&event); // On doit utiliser PollEvent car il ne faut pas attendre d'évènement
70         de l'utilisateur pour mettre à jour la fenêtre
71         switch(event.type)
72         {
```

```

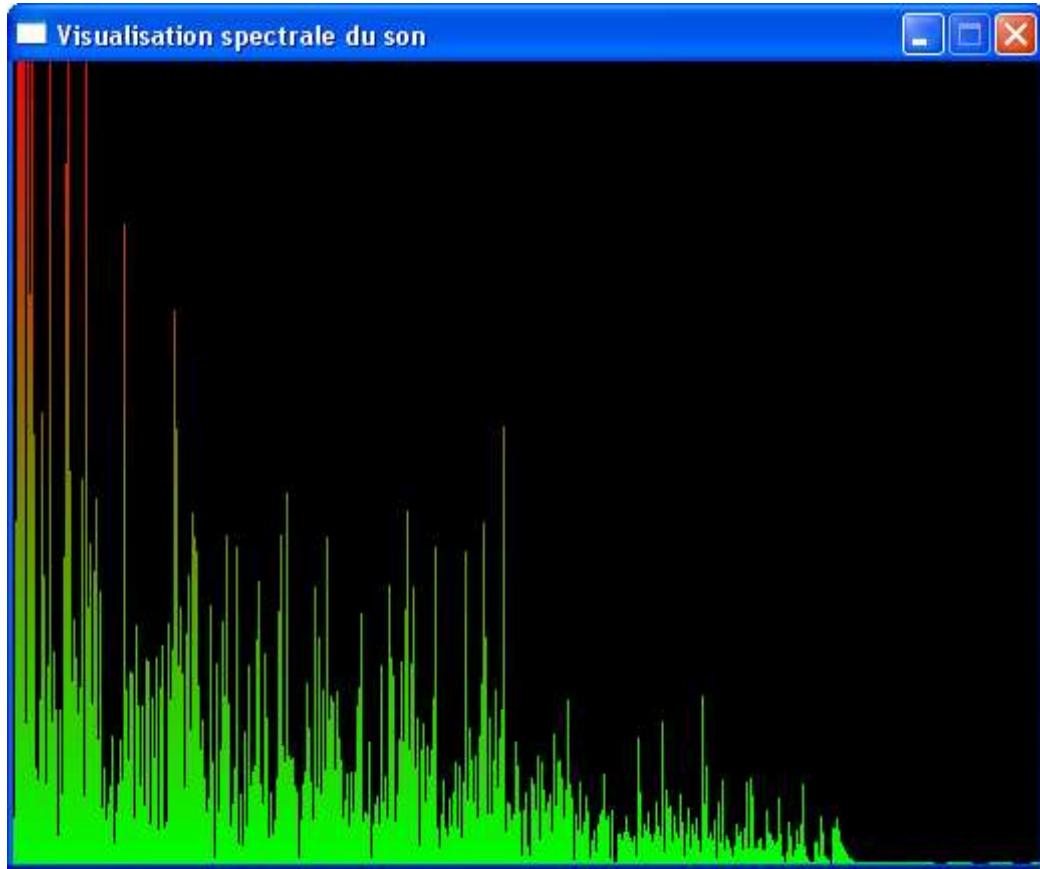
71     case SDL_QUIT:
72         continuer = 0;
73         break;
74     }
75
76     /* On efface l'écran à chaque fois avant de dessiner le graphe (fond noir) */
77     SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 0, 0, 0));
78
79     /* Gestion du temps
80     -----
81     On compare le temps actuel par rapport au temps précédent (dernier passage dans la
boucle)
82     Si ça fait moins de 25 ms (DELAI_RAFRAICHISSEMENT)
83     Alors on attend le temps qu'il faut pour qu'au moins 25 ms se soient écoulées.
84     On met ensuite à jour tempsPrecedent avec le nouveau temps */
85
86     tempsActuel = SDL_GetTicks();
87     if (tempsActuel - tempsPrecedent < DELAI_RAFRAICHISSEMENT)
88     {
89         SDL_Delay(DELAI_RAFRAICHISSEMENT - (tempsActuel - tempsPrecedent));
90     }
91     tempsPrecedent = SDL_GetTicks();
92
93     /* Dessin du spectre sonore
94     -----
95
96     C'est la partie la plus intéressante. Il faut réfléchir un peu à la façon de dessiner
pour y arriver, mais c'est tout à fait faisable (la preuve).
97
98     On remplit le tableau de 512 floats via FMOD_Channel_GetSpectrum()
99     On travaille ensuite pixel par pixel sur la surface ecran pour dessiner les barres.
100    On fait une première boucle pour parcourir la fenêtre en largeur.
101    La seconde boucle parcourt la fenêtre en hauteur pour dessiner chaque barre.
102 */
103
104    /* On remplit le tableau de 512 floats. J'ai choisi de m'intéresser à la sortie gauche */
105    FMOD_Channel_GetSpectrum(canal, spectre, TAILLE_SPECTRE, 0, FMOD_DSP_FFT_WINDOW_RECT);
106
107    SDL_LockSurface(ecran); /* On bloque la surface ecran car on va directement modifier ses
pixels */
108
109    /* BOUCLE 1 : on parcourt la fenêtre en largeur (pour chaque barre verticale) */
110    for (i = 0 ; i < LARGEUR_FENETRE ; i++)
111    {
112        /* On calcule la hauteur de la barre verticale qu'on va dessiner.
113           spectre[i] nous renvoie un nombre entre 0 et 1 qu'on multiplie par 20 pour zoomer
afin de voir un peu mieux (comme je vous avais dit).
114           On multiplie ensuite par HAUTEUR_FENETRE pour que la barre soit agrandie par rapport
à la taille de la fenêtre. */
115        hauteurBarre = spectre[i] * 20 * HAUTEUR_FENETRE;
116
117        /* On vérifie que la barre ne dépasse pas la hauteur de la fenêtre
118           Si tel est le cas on coupe la barre au niveau de la hauteur de la fenêtre. */
119        if (hauteurBarre > HAUTEUR_FENETRE)
120            hauteurBarre = HAUTEUR_FENETRE;
121
122        /* BOUCLE 2 : on parcourt en hauteur la barre verticale pour la dessiner */
123        for (j = HAUTEUR_FENETRE - hauteurBarre ; j < HAUTEUR_FENETRE ; j++)

```

```
124     {
125         /* On dessine chaque pixel de la barre à la bonne couleur.
126            On fait simplement varier le rouge et le vert, chacun dans un sens différent.
127
128            j ne varie pas entre 0 et 255 mais entre 0 et HAUTEUR_FENETRE.
129            Si on veut l'adapter proportionnellement à la hauteur de la fenêtre, il suffit de
130            faire le calcul j / RATIO, où RATIO vaut (HAUTEUR_FENETRE / 255.0).
131            J'ai dû réfléchir 2-3 minutes pour trouver le bon calcul à faire, mais c'est du
132            niveau de tout le monde. Il suffit de réfléchir un tout petit peu */
133         setPixel(ecran, i, j, SDL_MapRGB(ecran->format, 255 - (j / RATIO), j / RATIO, 0));
134     }
135     }
136
137     SDL_UnlockSurface(ecran); /* On a fini de travailler sur l'écran, on débloque la surface */
138
139     SDL_Flip(ecran);
140 }
141
142 /* Le programme se termine.
143    On libère la musique de la mémoire
144    et on ferme FMOD et SDL */
145
146 FMOD_Sound_Release(musique);
147 FMOD_System_Close(system);
148 FMOD_System_Release(system);
149
150     SDL_Quit();
151 }
152
153 /* La fonction setPixel permet de dessiner pixel par pixel dans une surface */
154 void setPixel(SDL_Surface *surface, int x, int y, Uint32 pixel)
155 {
156     int bpp = surface->format->BytesPerPixel;
157
158     Uint8 *p = (Uint8 *)surface->pixels + y * surface->pitch + x * bpp;
159
160     switch(bpp) {
161     case 1:
162         *p = pixel;
163         break;
164
165     case 2:
166         *(Uint16 *)p = pixel;
167         break;
168
169     case 3:
170         if(SDL_BYTEORDER == SDL_BIG_ENDIAN) {
171             p[0] = (pixel >> 16) & 0xff;
172             p[1] = (pixel >> 8) & 0xff;
173             p[2] = pixel & 0xff;
174         } else {
175             p[0] = pixel & 0xff;
176             p[1] = (pixel >> 8) & 0xff;
177             p[2] = (pixel >> 16) & 0xff;
178         }
179         break;
180 }
```

```
180  
181     case 4:  
182         *(UInt32 *)p = pixel;  
183         break;  
184     }  
185 }
```

Vous devriez obtenir un résultat correspondant à la figure suivante.



Bien entendu, il vaut mieux une animation pour apprécier ce résultat. C'est donc ce que je vous propose de visualiser.

#### [Voir l'animation "Visualisation spectrale du son" \(4,3 Mo\)](#)

Notez que la compression a réduit la qualité du son et le nombre d'images par seconde. Le mieux est encore de télécharger le programme complet (avec son code source) pour tester chez soi. Vous pourrez ainsi apprécier le programme dans les meilleures conditions. 😊

#### [Télécharger le projet Code::Blocks complet + l'exécutable Windows \(335 Ko\)](#)

Il faut impérativement que le fichier `Hype_Home.mp3` soit placé dans le dossier du programme pour qu'il fonctionne (sinon il s'arrêtera tout de suite).

## Idées d'amélioration

Il est toujours possible d'améliorer un programme. Ici, j'ai par exemple des tonnes d'idées d'extensions qui pourraient aboutir à la création d'un véritable petit lecteur MP3.

- Il serait bien qu'on puisse choisir le MP3 qu'on veut lire. Il faudrait par exemple lister tous les .mp3 présents dans le dossier du programme. Nous n'avons pas vu comment faire ça, mais vous pouvez le découvrir par vous-mêmes. Un indice : utilisez la librairie dirent (il faudra inclure dirent.h). À vous de chercher des informations sur le web pour savoir comment l'utiliser.
- Si votre programme était capable de lire et gérer les playlists, ça serait encore mieux. Il existe plusieurs formats de playlist, le plus connu est le format M3U.
- Vous pourriez afficher le nom du MP3 en cours de lecture dans la fenêtre (il faudra utiliser SDL\_ttf).
- Vous pourriez afficher un indicateur pour qu'on sache où en est la lecture du morceau, comme cela se fait sur la plupart des lecteurs MP3.
- Vous pourriez aussi proposer de modifier le volume de lecture.
- etc.

Bref, il y a beaucoup à faire. Vous avez la possibilité de créer de beaux lecteurs, il ne tient plus qu'à vous de les coder !



INDIQUER QUE CE CHAPITRE N'EST PAS TERMINÉ



JOUER DU SON AVEC FMOD



QUIZ : QUIZ 3

## Le professeur

**Mathieu Nebra**

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

## Découvrez aussi ce cours en...



Livre



PDF

**OpenClassrooms**

[L'entreprise](#)[Alternance](#)[Forum](#)[Blog](#)[Nous rejoindre](#)

---

## Entreprises

[Employeurs](#)

---

## En plus

[Devenez mentor](#)[Aide et FAQ](#)[Conditions Générales d'Utilisation](#)[Politique de Protection des Données Personnelles](#)[Nous contacter](#)

---

[Français](#)