Annexe

Index

Piano Hero 2.0

Main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL.h>
#include <SDL/SDL_ttf.h>
#include <fmod.h>


#include "intro.c" //Allez savoir pourquoi lorqu'on met les autres .c dans
le projet ca plante
#include "selection.c"
#include "credit.c"
#include "tuto.c"
#include "score.c"
#include "jeu.c"

void accueil(SDL_Surface* ecran); void merci(SDL_Surface* ecran);
//Prototypes des fontions menu et merci (pas utile de faire un .h)


/*************************************************************************
*************************************************************************
**********

                                MAIN

        ------------------------------------------------------------------

    - On crée la fenetre de 1250x833
    - On associe l'icone
    - On donne le nom au programme
    - On lance l'intro puis le menu d'accueil
    - Lorsque l'on quitte le menu d'accueil et donc le jeu on affiche
"merci d'avoir joué" et on ferme la fenêtre


                                                                Auteur
: Jean

*************************************************************************
*************************************************************************
**********/


int main(int argc, char *argv[]) //On initialise juste la fenetre
{
    SDL_Surface *ecran = NULL; //init ecran
    ecran = SDL_SetVideoMode(1250, 833, 16, SDL_HWSURFACE|SDL_DOUBLEBUF);
// La fenêtre sera de 1250/833 pixels en 16bits
    SDL_WM_SetIcon(SDL_LoadBMP("images/icone.bmp"), NULL);//icone
    SDL_WM_SetCaption("Piano Hero 2.0", NULL);//nom de la fenetre
    intro(ecran);
    accueil(ecran);
```

```
    merci(ecran);
    SDL_Quit();
    return 0;
}


/*************************************************************************
*************************************************************************
**********

                                FONCTION ACCUEIL

        ------------------------------------------------------------------

    Affiche l'image de menu
    On a le choix entre 3 possibilités/fonctions
    On peut utiliser la souris et le clavier pour naviguer dans le menu,
        pour cela on a un curseur, on utilise sa position pour savoir
quelle
        fonction lancer

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main


                                                                    Auteur
: Jean

*************************************************************************
*************************************************************************
**********/


void accueil(SDL_Surface* ecran)
{
    //Menu
    SDL_Surface*menu = NULL,*selection=NULL;

    SDL_Event event;
    SDL_Rect positionMenu,positionSelection;
    positionMenu.x = 0; positionMenu.y = 0; positionSelection.x = 15;
positionSelection.y = 100;
    menu = SDL_LoadBMP("images/menu.bmp"); selection =
SDL_LoadBMP("images/selection.bmp");
    SDL_SetColorKey(selection, SDL_SRCCOLORKEY, SDL_MapRGB(selection-
>format, 0,0, 255));//transparence
    int continuer =
1,xmouse=0,ymouse=0,compteur=0,choix=0,difficulte=1,ArduinoClavier=0;
    while (continuer)
    {


SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255, 255));
SDL_BlitSurface(menu, NULL, ecran, &positionMenu);
SDL_BlitSurface(selection, NULL, ecran, &positionSelection);
SDL_WaitEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
```

```
case SDL_KEYDOWN: // Si appui sur une touche du clavier

    switch(event.key.keysym.sym)
    {
        case SDLK_UP:
            if (positionSelection.y > 100)
positionSelection.y=positionSelection.y-100;
        break;
        case SDLK_DOWN:
            if (positionSelection.y < 300)
positionSelection.y=positionSelection.y+100;
        break;
        case SDLK_ESCAPE:
            continuer=0;
        break;
        case SDLK_RETURN:
            if(positionSelection.y==100) {
                choix = mselection(ecran);
                difficulte=difficult(ecran);
                ArduinoClavier=clavierArduino(ecran);
                compteur=jeu(ecran,choix,difficulte,ArduinoClavier);
                score(ecran,compteur);}//on lance la partie puis on affiche
le score
            if(positionSelection.y==200) tuto(ecran);
            if(positionSelection.y==300) credit(ecran);
        break;
        default:
        break;
    }
break;

case SDL_MOUSEMOTION : //souris
SDL_GetMouseState( &xmouse, &ymouse );
if((xmouse<400)&&(ymouse<150)&&(ymouse>50)) positionSelection.y=100;
if((xmouse<400)&&(ymouse<250)&&(ymouse>150)) positionSelection.y=200;
if((xmouse<400)&&(ymouse<350)&&(ymouse>250)) positionSelection.y=300;
break;
case SDL_MOUSEBUTTONDOWN:
if (positionSelection.y==100) {
                choix = mselection(ecran);
                difficulte=difficult(ecran);
                ArduinoClavier=clavierArduino(ecran);
                compteur=jeu(ecran,choix,difficulte,ArduinoClavier);
                score(ecran,compteur);}//on lance la partie puis on affiche
le score
if (positionSelection.y==200) tuto(ecran);
if (positionSelection.y==300) credit(ecran);

break;
}



SDL_Flip(ecran);
    }
}


/*************************************************************************
*************************************************************************
**********
```

```
                                FONCTION MERCI

        ------------------------------------------------------------------

    Affiche l'image avec "merci d'avoir joué" pendant 1sec

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main


                                                                    Auteur
: Aurele

*******************************************************************************
*******************************************************************************
**********/
void merci(SDL_Surface* ecran)
{
    SDL_ShowCursor(SDL_DISABLE);//On n'affiche plus le curseur
    SDL_Surface *imageDeFond = NULL;
    imageDeFond = SDL_LoadBMP("images/merci.bmp");//on indique ou est
l'image de fond
    SDL_Rect positionFond;
    positionFond.x = 0;
    positionFond.y = 0;

    //Timer
int tempsDebut=SDL_GetTicks();//SDL_GetTicks donne le temps qu'il s'est
écoulé depuis le lancement du programme, on retire donc le temps qu'il
s'est écoulé entre le lancement de la fonction
int tempsActuel=0;
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255, 255));
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
SDL_Flip(ecran);

int continuer = 1;
while (continuer)
    {

tempsActuel = SDL_GetTicks() - tempsDebut; //temps
if (tempsActuel >= 1000) continuer=0;

    }


}
```

Crédit.c

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL.h>
#include <SDL/SDL_ttf.h>
#include <fmod.h>
```

```
/*****************************************************************************
*****************************************************************************
**********

                                FONCTION CREDIT

        ---------------------------------------------------------------

    On affiche l'image de fond sur lequel on fait monter les noms avec la
    meme fonction que celle qui fait descendre les notes (en fonction du
temps)
    On joue aussi de la musique

        ---------------------------------------------------------------
    ecran : fenêtre initialisée dans le main


                                                                Auteur
: Aurele

*****************************************************************************
*****************************************************************************
**********/

void credit(SDL_Surface* ecran)
{

SDL_ShowCursor(SDL_DISABLE);//On n'affiche plus le curseur
SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *Noms = NULL;
SDL_Rect positionFond,positionNoms;

positionFond.x = 0;
positionFond.y = 0;
positionNoms.x = 400;
positionNoms.y = 900;

imageDeFond = SDL_LoadBMP("images/credit.bmp");//on indique ou est l'image
de fond
Noms = SDL_LoadBMP("images/noms.bmp");
SDL_SetColorKey(Noms, SDL_SRCCOLORKEY, SDL_MapRGB(Noms->format, 0,0,
255));//transparence


    /*Musique*/
FMOD_SYSTEM *system;
FMOD_SOUND *musique;
FMOD_RESULT resultat;
FMOD_System_Create(&system);
FMOD_System_Init(system, 1, FMOD_INIT_NORMAL, NULL);
/* On ouvre la musique */
resultat = FMOD_System_CreateSound(system, "musique.mp3",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);

/* On joue la musique */
FMOD_System_PlaySound(system, FMOD_CHANNEL_FREE, musique, 0,NULL);

int tempsPrecedent = 0, tempsActuel = 0; //Timer
int tempsDebut=SDL_GetTicks();//SDL_GetTicks donne le temps qu'il s'est
écoulé depuis le lancement du programme, on retire donc le temps qu'il
s'est écoulé entre le lancement et le début du morceau
```

```
int continuer = 1;


while (continuer)
    {

            /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On remet le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

tempsActuel = SDL_GetTicks() - tempsDebut; //temps

 if (positionNoms.y>30) positionNoms.y= positionNoms.y-
tempsActuel/20+tempsPrecedent/20; // Si le crédit n'est toujours pas en
haut, faire remonter (meme principe que pour la note : on utilise le temps)
 tempsPrecedent = tempsActuel;
SDL_BlitSurface(Noms, NULL, ecran, &positionNoms);

    SDL_PollEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: /* Si appui sur une touche */

    switch(event.key.keysym.sym)
    {
        case SDLK_ESCAPE:
        continuer=0;
        break;
        default:
        break;
    }
break;

}



/* On met à jour l'affichage */
SDL_Flip(ecran);

    }


////////////////////////////////////////
/*Sortie de boucle*/
SDL_ShowCursor(SDL_ENABLE);//on réaffiche le curseur pour le menu
//Libération de l'espace
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Noms);
//pour la musique
FMOD_Sound_Release(musique);
FMOD_System_Close(system);
FMOD_System_Release(system);
}
```

Intro.c

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL_video.h>
#include <SDL/SDL.h>
#include <fmod.h>


/*************************************************************************
*************************************************************************
**********

                                FONCTION INTRO

        ------------------------------------------------------------------

    On affiche une image d'intro et on joue de la musique puis après 3sec
on affiche l'image avec
    écrit PIANO HERO avec le texte qui clignote en fondu
    On attend un evenement (souris ou clavier) si l'evenement est fait
     on sort de la fonction (on lance donc l'ecran d'accueil)

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main


                                                                    Auteur
: Jean

*************************************************************************
*************************************************************************
**********/

void intro(SDL_Surface* ecran)
{

SDL_ShowCursor(SDL_DISABLE);//On n'affiche plus le curseur
SDL_Event event; //pour pouvoir gerer les events
SDL_Surface*Intro = NULL,*Intro1 = NULL,*Appuie = NULL;
SDL_Rect positionIntro, positionAppuie;
positionIntro.x = 0; positionIntro.y = 0; positionAppuie.x = 370;
positionAppuie.y = 680; // positions
Intro = SDL_LoadBMP("images/intro.bmp");  Intro1 =
SDL_LoadBMP("images/intro1.bmp"); Appuie =
SDL_LoadBMP("images/appuie.bmp"); //chargement des images
int tempsActuel = 0;
int alpha = SDL_ALPHA_OPAQUE, i=0 ; //clignoter (en transparence)

     /*Musique*/
FMOD_SYSTEM *system;
FMOD_SOUND *musique;
FMOD_RESULT resultat;
FMOD_System_Create(&system);
FMOD_System_Init(system, 1, FMOD_INIT_NORMAL, NULL);
/* On ouvre la musique */
resultat = FMOD_System_CreateSound(system, "intro.mp3",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);

/* On joue la musique */
```

```
FMOD_System_PlaySound(system, FMOD_CHANNEL_FREE, musique, 0,NULL);


int continuer = 1;


while (continuer)
    {

    tempsActuel = SDL_GetTicks(); //timer

            /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir

    /*Blink (i correspond à si il a été totalement transparent ou non)*/
if (i==0) alpha=alpha-5; if (alpha ==255) i=0;
if (i==1) alpha=alpha+5; if (alpha ==0) i=1;

SDL_SetAlpha( Appuie, SDL_SRCALPHA | SDL_RLEACCEL, alpha ); //On applique
la transparence

////////////////////////////////////////////////////////////////////////////
///////////////


if (tempsActuel <=3000) SDL_BlitSurface(Intro1, NULL, ecran,
&positionIntro);
if (tempsActuel >=3000) { SDL_BlitSurface(Intro, NULL, ecran,
&positionIntro); SDL_BlitSurface(Appuie, NULL, ecran, &positionAppuie); }


SDL_PollEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: /* Si appui sur une touche */

    switch(event.key.keysym.sym)
    {
        case SDLK_ESCAPE:
        exit(EXIT_FAILURE);
        break;
        default:
        continuer=0;
        break;
    }
break;

case SDL_MOUSEBUTTONDOWN:
continuer=0;
break;

}


/* On met à jour l'affichage */
SDL_Flip(ecran);
```

```
    }
SDL_ShowCursor(SDL_ENABLE);//on réaffiche le curseur pour le menu
    //Libération de l'espace
        //images
SDL_FreeSurface(Intro);
SDL_FreeSurface(Intro1);
        //pour la musique
FMOD_Sound_Release(musique);
FMOD_System_Close(system);
FMOD_System_Release(system);

}
```

Jeu.c

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL.h>
#include <fmod.h>
#include <SDL/SDL_ttf.h>
#include <string.h>

#define TAILLE_MAX 10000 // Tableau de taille 10 000
#include <windows.h>
#include <conio.h>



/*============================================================================
====
  Définition de constantes
============================================================================
==*/
#define RX_SIZE         4096    /* taille tampon d'entrée  */
#define TX_SIZE         4096    /* taille tampon de sortie */
#define MAX_WAIT_READ   5000    /* temps max d'attente pour lecture (en ms)
*/


/*============================================================================
====
  Variables globales.
============================================================================
==*/
/* Handle du port COM ouvert */
HANDLE g_hCOM = NULL;

/* Délais d'attente sur le port COM */
COMMTIMEOUTS g_cto =
{
    MAX_WAIT_READ,  /* ReadIntervalTimeOut          */
    0,              /* ReadTotalTimeOutMultiplier   */
    MAX_WAIT_READ,  /* ReadTotalTimeOutConstant     */
    0,              /* WriteTotalTimeOutMultiplier  */
```

```
    0               /* WriteTotalTimeOutConstant    */
};

/* Configuration du port COM */
DCB g_dcb =
{
    sizeof(DCB),        /* DCBlength            */
    9600,               /* BaudRate             */
    TRUE,               /* fBinary              */
    FALSE,              /* fParity              */
    FALSE,              /* fOutxCtsFlow         */
    FALSE,              /* fOutxDsrFlow         */
    DTR_CONTROL_ENABLE, /* fDtrControl          */
    FALSE,              /* fDsrSensitivity      */
    FALSE,              /* fTXContinueOnXoff    */
    FALSE,              /* fOutX                */
    FALSE,              /* fInX                 */
    FALSE,              /* fErrorChar           */
    FALSE,              /* fNull                */
    RTS_CONTROL_ENABLE, /* fRtsControl          */
    FALSE,              /* fAbortOnError        */
    0,                  /* fDummy2              */
    0,                  /* wReserved            */
    0x100,              /* XonLim               */
    0x100,              /* XoffLim              */
    8,                  /* ByteSize             */
    NOPARITY,           /* Parity               */
    ONESTOPBIT,         /* StopBits             */
    0x11,               /* XonChar              */
    0x13,               /* XoffChar             */
    '?',                /* ErrorChar            */
    0x1A,               /* EofChar              */
    0x10                /* EvtChar              */
};

/*============================================================================
====
  Fonctions du module.
============================================================================
==*/
BOOL OpenCOM    (int nId);
BOOL CloseCOM   ();
BOOL ReadCOM    (void* buffer, int nBytesToRead, int* pBytesRead);

/*****************************************************************************
**********************************************************************
**********

                                FONCTION ARDUINO

        ---------------------------------------------------------------

Lorsque elle est appelée avec le parametre 1, cette fonction permet
d'ouvir le port serie.
Puis elle permet de récuperer la chaine de caractere envoyée par arduino,
de la transformer  en un nombre et de renvoyer ce nombre a la fonction jeu.

                                                                Auteur
: Yacine Saoudi
```

```
**********************************************************************
**********************************************************************
**********/
int arduino(int verif)
{


    char buffer[30];  //permettra de stocker la chaine de caracacteres
envoyée par la carte arduino
    int nId=0; //numero du port COM  a ouvrir (varie en fonction du pc ou
du port usb utilisé
    int succes=0;
    int nBytesRead = 0;
/* ouvrir le port COM est assez long (2-3 secondes), on doit donc l'ouvrir
une seule fois (au debut du jeu).
On n'appelle la fonction arduino avec 1 comme parametre que la premiere
fois, pour ouvrir le port COM.
 la partie suivante ne s'execute donc qu'une seule fois */
if (verif==1)
{
   while(succes == 0)
  {
     if(!OpenCOM(nId)) //si le port COM ne s'ouvre pas, on incremente nId
et on tentera donc d'ouvrir le port COM suivant au prochain tour de boucle
     {
        nId++;
     }
     else
     {
        succes=1; //si le port est ouvert on met fin a la boucle
     }


     if(nId>20) //si on a deja essayé les 20 premiers ports series, on
abandonnne car aucune manette n'est branchée.
     {
        succes=1;
     }

  }
}

        int buffernum = 0;


            if(ReadCOM(buffer, sizeof(buffer)-1, &nBytesRead)) //recupere
la chaine de caractere envoyée par l'arduino dans buffer
            {
            buffernum=atoi(buffer); //permet de transformer le buffer
(chaine de caractère) en un int afin de pouvoir le renvoyer a la fonction
jeu directement.
            }
            else
            {
                // printf("Erreur lors de la réception.\r\n");
            }

            return(buffernum);
   }
```

```
/*****************************************************************************
****
  OpenCOM : ouverture et configuration du port COM.
  entrée : nId : Id du port COM à ouvrir.
  retour : vrai si l'opération a réussi, faux sinon.
*****************************************************************************
***/
BOOL OpenCOM(int nId)
{
    /* variables locales */
    char szCOM[16];

    /* construction du nom du port, tentative d'ouverture */
    sprintf(szCOM, "COM%d", nId);
    g_hCOM = CreateFile(szCOM, GENERIC_READ|GENERIC_WRITE, 0, NULL,
                        OPEN_EXISTING, FILE_ATTRIBUTE_SYSTEM, NULL);
    if(g_hCOM == INVALID_HANDLE_VALUE)
    {
       // printf("Erreur lors de l'ouverture du port COM%d", nId);
        return FALSE;
    }

    /* affectation taille des tampons d'émission et de réception */
    SetupComm(g_hCOM, RX_SIZE, TX_SIZE);

    /* configuration du port COM */
    if(!SetCommTimeouts(g_hCOM, &g_cto) || !SetCommState(g_hCOM, &g_dcb))
    {
        //printf("Erreur lors de la configuration du port COM%d", nId);
        CloseHandle(g_hCOM);
        return FALSE;
    }

    /* on vide les tampons d'émission et de réception, mise à 1 DTR */
    PurgeComm(g_hCOM,
PURGE_TXCLEAR|PURGE_RXCLEAR|PURGE_TXABORT|PURGE_RXABORT);
    EscapeCommFunction(g_hCOM, SETDTR);
    return TRUE;
}

/*****************************************************************************
****
  CloseCOM : fermeture du port COM.
  retour : vrai si l'opération a réussi, faux sinon.
*****************************************************************************
***/
BOOL CloseCOM()
{
    /* fermeture du port COM */
    CloseHandle(g_hCOM);
    return TRUE;
}

/*****************************************************************************
****
  ReadCOM : lecture de données sur le port COM.
  entrée : buffer       : buffer où mettre les données lues.
           nBytesToRead : nombre max d'octets à lire.
           pBytesRead   : variable qui va recevoir le nombre d'octets lus.
```

```
  retour : vrai si l'opération a réussi, faux sinon.
--------------------------------------------------------------------------
----
  Remarques : - la constante MAX WAIT READ utilisée dans la structure
                COMMTIMEOUTS permet de limiter le temps d'attente si aucun
                caractères n'est présent dans le tampon d'entrée.
              - la fonction peut donc retourner vrai sans avoir lu de
données.
**************************************************************************
***/
BOOL ReadCOM(void* buffer, int nBytesToRead, int* pBytesRead)
{
    return ReadFile(g_hCOM, buffer, nBytesToRead, pBytesRead, NULL);
}




/**************************************************************************
**************************************************************************
**********

                                FONCTION JEU

        ------------------------------------------------------------------

    Fait tourner le jeu :
        = Lance la musique
        = Lit la "partition"
        =Boucle jeu :
            = Fais descendre les notes en fonction du temps et grâce au
temps (SDL_GetTicks)
            = Gestion des evenements (arduino si on a selectionné la
possibilité de jouer avec + clavier)
            = Gestion echec réussite
            = Affiche texte (nom morceau + score)
        = Comptabilise le pourcentage de notes réussies

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main
    choix : choix du morceau
    difficulté : choix de la difficulté
    ArduinoClavier : jouer avec le clavier ou avec Arduino
    RETURN => Pourcentage de réussite


                                                                Auteur
: Jean

**************************************************************************
**************************************************************************
**********/

int jeu(SDL_Surface* ecran,int choix,int difficulte,int ArduinoClavier)
{
```

```
SDL_Event event; //pour pouvoir gerer les events
SDL_ShowCursor(SDL_DISABLE);//On n'affiche plus le curseur
SDL_Surface *imageDeFond = NULL, *Note = NULL, *Note_do = NULL, *Note_re =
NULL, *Note_mi = NULL, *Note_fa = NULL, *Note_sol = NULL, *Note_la = NULL,
*Note_si = NULL; //Initialisation des images : on crée un pointeur pour
chaque image auquel on met la valeur NULL
SDL_Rect positionFond,positionNote[TAILLE_MAX],
positionNote_do,positionNote_re,positionNote_mi,positionNote_fa,positionNot
e_sol,positionNote_la,positionNote_si; //Initialisation des positions des
images

//Initialisation positions x et y
int i; for(i=0;i<TAILLE_MAX;i++)
        {positionNote[i].x=0; positionNote[i].y=0;} //Initialisation de
TOUTES les notes du morceau

positionNote_do.y = positionNote_re.y = positionNote_mi.y =
positionNote_fa.y = positionNote_sol.y = positionNote_la.y =
positionNote_si.y = 658;
positionNote_do.x = 265;
positionNote_re.x = 408;
positionNote_mi.x = 548;
positionNote_fa.x = 690;
positionNote_sol.x = 834;
positionNote_la.x = 973;
positionNote_si.x = 1117;
positionFond.x = 0;
positionFond.y = 0;


imageDeFond = SDL_LoadBMP("images/fond.bmp");//on indique ou est l'image de
fond


//Images
Note = SDL_LoadBMP("images/notes/note.bmp") ;
Note_do = SDL_LoadBMP("images/notes/do.bmp");
Note_re = SDL_LoadBMP("images/notes/re.bmp");
Note_mi = SDL_LoadBMP("images/notes/mi.bmp");
Note_fa = SDL_LoadBMP("images/notes/fa.bmp");
Note_sol = SDL_LoadBMP("images/notes/sol.bmp");
Note_la = SDL_LoadBMP("images/notes/la.bmp");
Note_si = SDL_LoadBMP("images/notes/si.bmp");
SDL_SetColorKey(Note, SDL_SRCCOLORKEY, SDL_MapRGB(Note->format, 0,0,
255));//transparence
//

 FILE* fichier = NULL;//pour lire la "partition"



/***********Musique***************
*******************************/

FMOD_SYSTEM *system;
FMOD_SOUND *musique;//musique :/
FMOD_RESULT resultat;
FMOD_System_Create(&system);
FMOD_System_Init(system, 1, FMOD_INIT_NORMAL, NULL);
```

```
/* On ouvre la musique en fonction du choix et on ouvre la partition qui
correspond (c'est gros pour pas grand chose)*/
switch(choix)
{
    case 0 :
    resultat = FMOD_System_CreateSound(system, "musiques/FrereJacques.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/FrereJacques.txt", "r"); //Le fichier texte
qui contient la "partition"
    break;
    case 1 :
    resultat = FMOD_System_CreateSound(system, "musiques/Muse.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Muse.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 2 :
    resultat = FMOD_System_CreateSound(system,
"musiques/AuClairDeLaLune.mid", FMOD_SOFTWARE | FMOD_2D |
FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/AuClairDeLaLune.txt", "r"); //Le fichier
texte qui contient la "partition"
    break;
    case 3 :
    resultat = FMOD_System_CreateSound(system, "musiques/Titanic.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Titanic.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 4 :
    resultat = FMOD_System_CreateSound(system, "musiques/MJ.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/MJ.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 5 :
    resultat = FMOD_System_CreateSound(system, "musiques/Clocks.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Clocks.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 6 :
    resultat = FMOD_System_CreateSound(system, "musiques/Laputa.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Laputa.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 7 :
    resultat = FMOD_System_CreateSound(system, "musiques/YMCA.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/YMCA.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 8 :
    resultat = FMOD_System_CreateSound(system, "musiques/Changes.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Changes.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 10 :
```

```
    resultat = FMOD_System_CreateSound(system,
"musiques/Dancing_in_the_Dark.mid", FMOD_SOFTWARE | FMOD_2D |
FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Dancing_in_the_Dark.txt", "r"); //Le fichier
texte qui contient la "partition"
    break;
    case 11 :
    resultat = FMOD_System_CreateSound(system, "musiques/Born_to_Run.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Born_to_Run.txt", "r"); //Le fichier texte
qui contient la "partition"
    break;
    case 12 :
    resultat = FMOD_System_CreateSound(system, "musiques/Bruno Mars.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Bruno Mars.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 13 :
    resultat = FMOD_System_CreateSound(system, "musiques/Bad day.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Bad day.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 15 :
    resultat = FMOD_System_CreateSound(system, "musiques/The Fray.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/The Fray.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 16 :
    resultat = FMOD_System_CreateSound(system, "musiques/Led Zep.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Led Zep.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 17 :
    resultat = FMOD_System_CreateSound(system, "musiques/Naruto.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Naruto.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    case 18 :
    resultat = FMOD_System_CreateSound(system, "musiques/Somebody to love -
Queen.mid", FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Somebody to love - Queen.txt", "r"); //Le
fichier texte qui contient la "partition"
    break;
    case 19 :
    resultat = FMOD_System_CreateSound(system, "musiques/Viva.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/Viva.txt", "r"); //Le fichier texte qui
contient la "partition"
    break;
    default:
    break;

}


   if (resultat != FMOD_OK)//verification que la musique marche
```

```
    {
        fprintf(stderr, "Impossible de lire le fichier audio.wav\n");
        exit(EXIT_FAILURE);
    }

//Bruit pour le fail (on est obligé de créer un autre systeme pour pas
arreter la musique)
FMOD_SYSTEM *systemf;
FMOD_SOUND *fail = NULL;
FMOD_System_CreateSound(systemf, "fail.wav", FMOD_CREATESAMPLE, 0, &fail);
FMOD_RESULT resultatf;

    /* Création et initialisation d'un objet système */
    FMOD_System_Create(&systemf);
    FMOD_System_Init(systemf, 1, FMOD_INIT_NORMAL, NULL);

    /* Chargement du son et vérification du chargement */
    resultatf = FMOD_System_CreateSound(system, "fail.wav",
FMOD_CREATESAMPLE, 0, &fail);
    if (resultatf != FMOD_OK)
    {
        fprintf(stderr, "Impossible de lire le fichier audio.wav\n");
        exit(EXIT_FAILURE);
    }

 /*****************************
 Initialisation pour le texte
 *****************************/
    char caracteres[20] = "",caracteres2[20] = ""; // Tableau de char
suffisamment grand pour le score
    TTF_Init(); //Initialisation de la banque de donnée pour le texte
    int compteur=0; //Pour le score
    SDL_Surface *score = NULL;
    SDL_Rect position;
    TTF_Font *police = NULL, *police2 = NULL; //TTF_OpenFont doit stocker
son résultat dans une variable de type TTF_Font
    SDL_Color couleurNoire = {0, 0, 0}; //couleur police => noir
    police = TTF_OpenFont("police.ttf", 70);//police choisie, taille police
    police2 = TTF_OpenFont("score.ttf", 70);//police choisie, taille police
//////////////////////////////////////////////

 /************************************************
        +Lecture du fichier texte (Yacine)
        +On met les notes a leur place (do,ré...)
            en fonction de la difficulté
 ************************************************/



char chaine[TAILLE_MAX]="";
int debut[TAILLE_MAX ];
int note[TAILLE_MAX ];
int tempsFin = 0 ;
 int j=0,compteurNotes=0;
 while (fgets(chaine, TAILLE_MAX, fichier) != NULL) //tant que le fichier
n'a pas été totalement parcouru (fgets s'incremente automatiquement)
   {

        ///Easy
       if (difficulte==0){
```

```
    if (compteurNotes%3==0)//Pour la difficulté, on ne prend qu'une note
sur 3 quand on a choisi l'option facile
            {
                sscanf(chaine, "%d - %d", &debut[j], &note[j]); // recupere
la note et la date

switch (note[j])//On met en place les notes
{

   case 0 : //do
        positionNote[j].x = 250;
        break;
   case 1 : //ré
        positionNote[j].x = 395;
        break;
   case 2 : //mi
        positionNote[j].x = 525;
        break;
   case 3 : //fa
        positionNote[j].x = 680;
        break;
   case 4 : //sol
        positionNote[j].x = 820;
        break;
   case 5 : //la
        positionNote[j].x = 960;
        break;
   case 6 : //si
        positionNote[j].x = 1100;
        break;
    case 8 : // Pour la fin
        tempsFin = debut[j];
        positionNote[j].x = 20000;//pour pas afficher la note
   default :
        break;

        }
        j++;
            }
            compteurNotes++;
            }

        ///Normal
       if (difficulte==1){
    if (compteurNotes%2==0)//Pour la difficulté, on ne prend qu'une note
sur deux quand on a choisi l'option facile
            {
                sscanf(chaine, "%d - %d", &debut[j], &note[j]); // recupere
la note et la date

switch (note[j])//On met en place les notes
{

   case 0 : //do
        positionNote[j].x = 250;
        break;
   case 1 : //ré
        positionNote[j].x = 395;
        break;
   case 2 : //mi
        positionNote[j].x = 525;
```

```
            break;
        case 3 : //fa
            positionNote[j].x = 680;
            break;
        case 4 : //sol
            positionNote[j].x = 820;
            break;
        case 5 : //la
            positionNote[j].x = 960;
            break;
        case 6 : //si
            positionNote[j].x = 1100;
            break;
         case 8 : // Pour la fin
            tempsFin = debut[j];
            positionNote[j].x = 20000;//pour pas afficher la note
        default :
            break;

            }
            j++;
                }
                compteurNotes++;
                }

            ///Difficile
            if (difficulte==2){
                //On prend toutes les notes
                        sscanf(chaine, "%d - %d", &debut[j], &note[j]); //
recupere la note et la date

switch (note[j])//On met en place les notes
{

        case 0 : //do
            positionNote[j].x = 250;
            break;
        case 1 : //ré
            positionNote[j].x = 395;
            break;
        case 2 : //mi
            positionNote[j].x = 525;
            break;
        case 3 : //fa
            positionNote[j].x = 680;
            break;
        case 4 : //sol
            positionNote[j].x = 820;
            break;
        case 5 : //la
            positionNote[j].x = 960;
            break;
        case 6 : //si
            positionNote[j].x = 1100;
            break;
         case 8 : // Pour la fin
            tempsFin = debut[j];
            positionNote[j].x = 20000;//pour pas afficher la note
        default :
            break;
```

```
        }
        j++;
            }


   }


 ////////////////////////////////////////////

int pourcent[TAILLE_MAX] = {0},pourcentFinal=0,totalNotes=0;//Pour le
pourcentage de réussite
int k=0;//notes 1,2,3....
int tempsDebut=SDL_GetTicks();//SDL_GetTicks donne le temps qu'il s'est
écoulé depuis le lancement du programme, on retire donc le temps qu'il
s'est écoulé entre le lancement et le début du morceau
int a=0,z=0,e=0,r=0,t=0,y=0,u=0;//notes
int tempsPrecedent = 0, tempsActuel = 0, tempsNote[7]; //Timer (temps note
permet de savoir a quel instant t la note a été jouée
int continuer = 1;

 //Boucle jeu

 if (ArduinoClavier) arduino(1);//On lance arduino pour avoir le numéro
port série qui correspond

while (continuer)
{


/* On joue la musique au bon moment de manière à ce qu'elle soit
synchronisée avec les notes qui défilent */
if ((tempsActuel>=2700)&&(tempsActuel<=2750))FMOD_System_PlaySound(system,
FMOD_CHANNEL_FREE, musique, 0,NULL);


    /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On remet le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

tempsActuel = SDL_GetTicks() - tempsDebut; //temps




/*********************************************
 + On affiche les notes
 + On fait descendre les notes le long des lignes
    J'ai galeré pour ca !
*********************************************/


    if(tempsActuel>=debut[k+1]) k++; // passage à la note suivante



int l=k;

    do
```

```
    {
            if (tempsActuel>=debut[0]) positionNote[l].y=
positionNote[l].y+tempsActuel/10*2-tempsPrecedent/10*2;//descente de la
note en utilisant le temps comme réference (elle met du coup 2.7sec a
desendre) (la condition corrige le bug de la 1ere note)

             if (positionNote[l].y>575) positionNote[l].y=10000; // si la
note arrive en bas on la fait "disparaitre"

        SDL_BlitSurface(Note, NULL, ecran, &positionNote[l]);//on affiche
les notes
        l--;
    }while(l>=0);



///La boucle do while est la car il faut la faire au moins une fois quand
il n'y a qu'un seule note
///Pour afficher les notes précédentes (sinon on a qu'une seule note
affichée)


/*Pour sortir du morceau à la fin*/
if (tempsActuel >= tempsFin) continuer = 0;



///


tempsPrecedent = tempsActuel; // comme on utilise le temps pour la boucle
d'avant on le met ici (on pourrait le mettre à la toute fin, ce qui serait
plus logique mais ici on comprend mieux)



/************************************************************
************************************************************
Fonction pour les touches
************************************************************
************************************************************/



/*******************************************************
Switch pour savoir quelle touche a étée enfoncée. Si
une touche est enfoncée on donne a la variable la valeur
1 et on enregistre le temps a laquelle la note a été
"jouée"
*******************************************************/

SDL_PollEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: /* Si appui sur une touche*/

    switch(event.key.keysym.sym)
    {
```

```
        case SDLK_q: //a
        a=1;
        tempsNote[0]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_w: //z
        z=1;
        tempsNote[1]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_e: //e
        e=1;
        tempsNote[2]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_r: //r
        r=1;
        tempsNote[3]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_t: //t
        t=1;
        tempsNote[4]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_y: //y
        y=1;
        tempsNote[5]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_u: //u
        u=1;
        tempsNote[6]=SDL_GetTicks() - tempsDebut;
        break;
        case SDLK_ESCAPE:
        continuer=0;
        break;
        default:
        break;
    }
break;

}



/*************************************************************
On a donné à une variable la valeur 1 si l'evenement a été
réalisé, pour que deux évenements soient pris en compte en même
temps on laisse l'action que provoque l'evenement durer 250ms
(on utlise le temps enregistré avant pour savoir quand 250ms
 sont écoulées)
ce qui donne l'impression que les evenements sont simultannés
Si la variable=0, l'action n'est pas réalisée
*************************************************************/
if (a)
    {
        SDL_BlitSurface(Note_do, NULL, ecran, &positionNote_do);
        if(tempsActuel-tempsNote[0]>250) a=0;
    }
if (z)
    {
        SDL_BlitSurface(Note_re, NULL, ecran, &positionNote_re);
        if(tempsActuel-tempsNote[1]>250) z=0;
    }
if (e)
```

```
    {
        SDL_BlitSurface(Note_mi, NULL, ecran, &positionNote_mi);
        if(tempsActuel-tempsNote[2]>250) e=0;
    }
if (r)
    {
        SDL_BlitSurface(Note_fa, NULL, ecran, &positionNote_fa);
        if(tempsActuel-tempsNote[3]>250) r=0;
    }
if (t)
    {
        SDL_BlitSurface(Note_sol, NULL, ecran, &positionNote_sol);
        if(tempsActuel-tempsNote[4]>250) t=0;
    }
if (y)
    {
        SDL_BlitSurface(Note_la, NULL, ecran, &positionNote_la);
        if(tempsActuel-tempsNote[5]>250) y=0;
    }
if (u)
    {
        SDL_BlitSurface(Note_si, NULL, ecran, &positionNote_si);
        if(tempsActuel-tempsNote[6]>250) u=0;
    }


/*****************************************************************
Cette partie permet de déterminer quels boutons poussoirs sont enfoncés a
partir du nombre
renvoyé par la fonction arduino.
Si un bouton est enfoncé on lui attribue un caractère propre, ainsi lors
des tests de succès
cette information sera gérée exactement comme les évènements au clavier.


Bugs  incompréhensibles liés a l'ajout d'arduino:
On ne peux plus quitter la partie en cours (parfois si)
Au début de certains morceaux  des notes bizarres s'affichent
mais cela revient a la normale après 2-3 secondes.


                        Auteur : Yacine Saoudi
*****************************************************************/

if (ArduinoClavier)
{

int arduinoIu=0;
arduinoIu=arduino(0);

char boutonArd[8];
sprintf(boutonArd, "%d", arduinoIu);
//on transforme le nombre stocké dans arduinoIu en chaine de caractère.

//cela permet de tester la présence ou non d'un caractère dans celle ci:
if(strstr(boutonArd, "1")!=NULL) a=1;
if(strstr(boutonArd, "2")!=NULL) z=1;
if(strstr(boutonArd, "3")!=NULL) e=1;
if(strstr(boutonArd, "4")!=NULL) r=1;
if(strstr(boutonArd, "5")!=NULL) t=1;
if(strstr(boutonArd, "6")!=NULL) y=1;
```

```
if(strstr(boutonArd, "7")!=NULL) u=1;

}


/*********************************************************************
Fonction réussit ou raté (si la note est "jouée" au bon moment)
*********************************************************************/


l=k;//on teste toutes les notes qui on étées affichées

    do
    {
        //réussi
        if
((a)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 250))  {compteur++; pourcent[l]=1;} //compteur => score
        if
((z)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 395))  {compteur++; pourcent[l]=1;} //pourcent => la note spécifique l
est réussie ou non (1= réussi)
        if
((e)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 525))  {compteur++; pourcent[l]=1;}
        if
((r)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 680))  {compteur++; pourcent[l]=1;}
        if
((t)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 820))  {compteur++; pourcent[l]=1;}
        if
((y)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 960))  {compteur++; pourcent[l]=1;}
        if
((u)&&(positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x
== 1100)) {compteur++; pourcent[l]=1;}
        //fail
        /*
        if
((a==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 250)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL);
        if
((z==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 395)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL);
        if
((e==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 525)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL);
        if
((r==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 680)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL);
        if
((t==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 820)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL);
```

```
        if
((y==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 960)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL);
        if
((u==0)&&(positionNote[l].y>560)&&(positionNote[l].y<570)&&(positionNote[l]
.x == 1100)) FMOD_System_PlaySound(systemf, FMOD_CHANNEL_FREE, fail, 0,
NULL); //son de fausse note
        */
        l--;
    }while(l>=0); //j'aime bien les do while (mêmes raisons qu'au dessus)



/*********************************
Fonction pour le texte, ici le score
et le nom du morceau
*********************************/

//En cas d'erreur (plus propre)
if(TTF_Init() == -1)
{

    fprintf(stderr, "Erreur d'initialisation de TTF_Init : %s\n",
TTF_GetError());
    exit(EXIT_FAILURE);
}

/*score*/


     /* Écriture du texte dans la SDL_Surface texte en mode Solid (car il
change souvent)*/
     sprintf(caracteres, "Score : %d", compteur);
     SDL_FreeSurface(score);//On efface la surface précédente (sinon ca
prend 2go de RAM)
     score = TTF_RenderText_Solid(police, caracteres, couleurNoire);

    //Position score//
        position.x = 20;
        position.y = 450;
        SDL_BlitSurface(score, NULL, ecran, &position); /* Blit du texte*/

/*titre*/

   /* Titre en fonction du choix */
   switch (choix)
   {
       case 0 :
       sprintf(caracteres2, "Frère Jacques");
       break;
       case 1 :
       sprintf(caracteres2, "Starlight - Muse");
       break;
       case 2 :
       sprintf(caracteres2, "Au Clair de la Lune");
       break;
       case 3 :
       sprintf(caracteres2, "Titanic");
       break;
       case 4 :
```

```
        sprintf(caracteres2, "Black or White - MJ");
        break;
        case 5 :
        sprintf(caracteres2, "Clocks - Coldplay");
        break;
        case 6 :
        sprintf(caracteres2, "Laputa");
        break;
        case 7 :
        sprintf(caracteres2, "YMCA");
        break;
        case 8 :
        sprintf(caracteres2, "Changes");
        break;
        case 10 :
        sprintf(caracteres2, "Dancing in the Dark");
        break;
        case 11 :
        sprintf(caracteres2, "Born to Run");
        break;
        case 12 :
        sprintf(caracteres2, "Just the way you are");
        break;
        case 13 :
        sprintf(caracteres2, "Bad day");
        break;
        case 15 :
        sprintf(caracteres2, "The Fray");
        break;
        case 16 :
        sprintf(caracteres2, "Starway to heaven");
        break;
        case 17 :
        sprintf(caracteres2, "Naruto");
        break;
        case 18 :
        sprintf(caracteres2, "Somebody to love - Queen");
        break;
        case 19 :
        sprintf(caracteres2, "Viva la Vida - Coldplay");
        break;
        default :
        break;
    }

     SDL_FreeSurface(score);//On efface la surface précédente (sinon ca
prend 2go de RAM)
     score = TTF_RenderText_Solid(police2, caracteres2, couleurNoire);

    //Position score//
        position.x = 20;
        position.y = 60;
        SDL_BlitSurface(score, NULL, ecran, &position); /* Blit du texte*/


////////////////////////////////////////////////////////////////////////
//////////////

/* On met à jour l'affichage */
SDL_Flip(ecran);
```

```
if (continuer==0) CloseCOM   ();//On ferme le port série
}

/*****************************************************
Fonction pour obtenir le pourcentage de réussite
On utilise la variable pourcent (faire une vraie fonction
                                 de ca serait facile mais
                                 inutile)
Bug a corriger : Si on finit pas le morceau le pourcentage
est faux (narmol)
*****************************************************/

for(totalNotes=0;totalNotes<k;totalNotes++)
{
    if (pourcent[totalNotes]==1) pourcentFinal++;
}
 pourcentFinal=pourcentFinal*100/totalNotes;//on fait le pourcentage (il
faut finir le morceau pour que le pourcentage soit juste)




//////////////////////////////////////////
/*Sortie de boucle*/
//////////////////////////////////////////
//Libération de l'espace
    //images
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Note);
SDL_FreeSurface(Note_do);
SDL_FreeSurface(Note_re);
SDL_FreeSurface(Note_mi);
SDL_FreeSurface(Note_fa);
SDL_FreeSurface(Note_sol);
SDL_FreeSurface(Note_la);
SDL_FreeSurface(Note_si);
    //pour la musique
FMOD_Sound_Release(musique);
FMOD_Sound_Release(fail);
FMOD_System_Close(system);
FMOD_System_Release(system);

//return compteur;//le score (le pourcentage est plus interessant)
return pourcentFinal;//le pourcentage de réussite
}
```

Selection.c

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL.h>
#include <SDL/SDL_ttf.h>
```

```
/*****************************************************************************
*****************************************************************************
**********

                                FONCTION MSELECTION

        ---------------------------------------------------------------------

    Permet de choisir le morceau, la méthode de séléction fonctionne comme
pour l'acceuil
    Si on choisit "-suivant-" au lieu de sortir de la boucle on lance le
deuxieme ecran de
    sélection de morceaux et si on sélectionne le morceau dans l'autre menu
la valeur retournée
    sera celle retournée par MSELECTION2
        ---------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main
    RETURN => La valeur qui correspond dans la fonction jeu au morceau
voullu


                                                                    Auteur
: Jean

*****************************************************************************
*****************************************************************************
**********/
int mselection(SDL_Surface* ecran)
{

SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *Selection = NULL;
SDL_Rect positionFond,positionSelection, positionsTitres [10];

positionFond.x = 0;
positionFond.y = 0;
positionSelection.x = 100;
positionSelection.y = 150;
int i=0,choix=0;
for (i=0;i<10;i++)
{
    if (i<5)
    {
    positionsTitres [i].x = 140;
    positionsTitres [i].y = 100+i*100;
    }
        if (i>=5)
    {
    positionsTitres [i].x = 700;
    positionsTitres [i].y = 100+(i-5)*100;
    }
}

imageDeFond = SDL_LoadBMP("images/credit.bmp"); Selection =
SDL_LoadBMP("images/selection.bmp");//on indique ou est l'image de fond
SDL_SetColorKey(Selection, SDL_SRCCOLORKEY, SDL_MapRGB(Selection->format,
0,0, 255));//transparence
int continuer = 1,xmouse=0,ymouse=0;
```

```
/**********************************
Fonction pour le texte
**********************************/
   char caracteres[10][20] = {""}; // Tableau de char suffisamment grand
pour les titres(tableau a 2 dimension, ca évite de faire 10 fois la meme
chose)
    TTF_Init(); //Initialisation de la banque de donnée pour le texte
    SDL_Surface *titre[10] = {NULL};
    TTF_Font *police = NULL; //TTF_OpenFont doit stocker son résultat dans
une variable de type TTF_Font
    SDL_Color couleurBlanc = {255, 255, 255}; //couleur police => noir
    police = TTF_OpenFont("score.ttf", 100);//police choisie, taille police

//En cas d'erreur (plus propre)
if(TTF_Init() == -1)
{

    fprintf(stderr, "Erreur d'initialisation de TTF_Init : %s\n",
TTF_GetError());
    exit(EXIT_FAILURE);
}


sprintf(caracteres[0], "Frère Jacques"); sprintf(caracteres[1], "Starlight
- Muse"); sprintf(caracteres[2], "Au clair de la Lune");
sprintf(caracteres[3], "Titanic");
sprintf(caracteres[4], "Black White - MJ"); sprintf(caracteres[5], "Clocks
- Coldplay"); sprintf(caracteres[6], "Laputa"); sprintf(caracteres[7],
"YMCA");
sprintf(caracteres[8], "Changes"); sprintf(caracteres[9], " - Suivant - ");
//tous les titres
for(i=0;i<10;i++) titre[i] = TTF_RenderText_Solid(police, caracteres[i],
couleurBlanc);



while (continuer)
    {

        choix=0;//Pour pouvoir reselectionner meme si on est allé dans le
menu suivant car sinon choix =14 et ca fait tout buguer

            /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On remet le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
SDL_BlitSurface(Selection, NULL, ecran, &positionSelection);

for(i=0;i<10;i++) SDL_BlitSurface(titre[i], NULL, ecran,
&positionsTitres[i]); /* Blit du texte*/



SDL_WaitEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
```

```
case SDL_KEYDOWN: /* Si appui sur une touche */

    switch(event.key.keysym.sym)
    {
        case SDLK_ESCAPE:
        continuer=0;
        break;
        case SDLK_RETURN:
        if ((positionSelection.y==550)&&(positionSelection.x==650))
choix=mselection2(ecran) ;
        if (choix ==14);
        else continuer = 0;
        break;
        case SDLK_UP:
            if (positionSelection.y > 150)
positionSelection.y=positionSelection.y-100;
        break;
        case SDLK_DOWN:
            if (positionSelection.y < 550)
positionSelection.y=positionSelection.y+100;
        break;
        case SDLK_RIGHT:
            if (positionSelection.x==100) positionSelection.x=650;
        break;
        case SDLK_LEFT:
            if (positionSelection.x==650) positionSelection.x=100;
        break;
        default:
        break;
    }
break;

case SDL_MOUSEMOTION : //souris
SDL_GetMouseState( &xmouse, &ymouse );
if((ymouse<200)&&(ymouse>50)) positionSelection.y=150;
if((ymouse<300)&&(ymouse>200)) positionSelection.y=250;
if((ymouse<400)&&(ymouse>300)) positionSelection.y=350;
if((ymouse<500)&&(ymouse>400)) positionSelection.y=450;
if((ymouse<600)&&(ymouse>500)) positionSelection.y=550;
if(xmouse<400) positionSelection.x=100;
if(xmouse>400) positionSelection.x=650;
break;
case SDL_MOUSEBUTTONDOWN:
if ((positionSelection.y==550)&&(positionSelection.x==650))
choix=mselection2(ecran) ; //Pour suivant
if (choix ==14);
else continuer = 0;
break;

}



/* On met à jour l'affichage */
SDL_Flip(ecran);

    }


//////////////////////////////////////////
```

```
/*Choix morceau*/
if ((positionSelection.y==150)&&(positionSelection.x==100)) choix=0;
if ((positionSelection.y==250)&&(positionSelection.x==100)) choix=1;
if ((positionSelection.y==350)&&(positionSelection.x==100)) choix=2;
if ((positionSelection.y==450)&&(positionSelection.x==100)) choix=3;
if ((positionSelection.y==550)&&(positionSelection.x==100)) choix=4;
if ((positionSelection.y==150)&&(positionSelection.x==650)) choix=5;
if ((positionSelection.y==250)&&(positionSelection.x==650)) choix=6;
if ((positionSelection.y==350)&&(positionSelection.x==650)) choix=7;
if ((positionSelection.y==450)&&(positionSelection.x==650)) choix=8;



//Libération de l'espace
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Selection);
return choix; //On retourne la valeur du morceau choisi
}



/*************************************************************************
*************************************************************************
**********

                                FONCTION MSELECTION2

        ------------------------------------------------------------------

        Meme fonction que MSELECTION mais retourne des valeurs differentes

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main
    RETURN => La valeur qui correspond dans la fonction jeu au morceau
voullu

                                                                  Auteur
: Jean

*************************************************************************
*************************************************************************
**********/
int mselection2(SDL_Surface* ecran)
{

SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *Selection = NULL;
SDL_Rect positionFond,positionSelection, positionsTitres [10];

positionFond.x = 0;
positionFond.y = 0;
positionSelection.x = 100;
positionSelection.y = 150;
int i=0,choix=0;
for (i=0;i<10;i++)
{
    if (i<5)
    {
    positionsTitres [i].x = 140;
```

```
    positionsTitres [i].y = 100+i*100;
    }
        if (i>=5)
    {
    positionsTitres [i].x = 700;
    positionsTitres [i].y = 100+(i-5)*100;
    }
}

imageDeFond = SDL_LoadBMP("images/credit.bmp"); Selection =
SDL_LoadBMP("images/selection.bmp");//on indique ou est l'image de fond
SDL_SetColorKey(Selection, SDL_SRCCOLORKEY, SDL_MapRGB(Selection->format,
0,0, 255));//transparence
int continuer = 1,xmouse=0,ymouse=0;



/*********************************
Fonction pour le texte
*********************************/
   char caracteres[10][21] = {""}; // Tableau de char suffisamment grand
pour les titres(tableau a 2 dimension, ca évite de faire 10 fois la meme
chose)
    TTF_Init(); //Initialisation de la banque de donnée pour le texte
    SDL_Surface *titre[10] = {NULL};
    TTF_Font *police = NULL; //TTF_OpenFont doit stocker son résultat dans
une variable de type TTF_Font
    SDL_Color couleurBlanc = {255, 255, 255}; //couleur police => noir
    police = TTF_OpenFont("score.ttf", 100);//police choisie, taille police

//En cas d'erreur (plus propre)
if(TTF_Init() == -1)
{

    fprintf(stderr, "Erreur d'initialisation de TTF_Init : %s\n",
TTF_GetError());
    exit(EXIT_FAILURE);
}

////////////////////////////////////////////////////////////////////////////
////
//2eme
page////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//
sprintf(caracteres[0], "Dancing in the Dark"); sprintf(caracteres[1], "Born
to Run"); sprintf(caracteres[2], "Just the way you are");
sprintf(caracteres[3], "Bad day");
sprintf(caracteres[4], "- Retour -"); sprintf(caracteres[5], "The Fray");
sprintf(caracteres[6], "Starway to heaven"); sprintf(caracteres[7],
"Naruto");
sprintf(caracteres[8], "Somebody to love"); sprintf(caracteres[9], "Viva la
Vida"); //tous les titres
for(i=0;i<10;i++) titre[i] = TTF_RenderText_Solid(police, caracteres[i],
couleurBlanc);



while (continuer)
    {
```

```
            /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On remet le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
SDL_BlitSurface(Selection, NULL, ecran, &positionSelection);

for(i=0;i<10;i++) SDL_BlitSurface(titre[i], NULL, ecran,
&positionsTitres[i]); /* Blit du texte*/



SDL_WaitEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: /* Si appui sur une touche */

    switch(event.key.keysym.sym)
    {
        case SDLK_ESCAPE:
        continuer=0;
        break;
        case SDLK_RETURN:
        continuer=0;
        break;
        case SDLK_UP:
            if (positionSelection.y > 150)
positionSelection.y=positionSelection.y-100;
        break;
        case SDLK_DOWN:
            if (positionSelection.y < 550)
positionSelection.y=positionSelection.y+100;
        break;
        case SDLK_RIGHT:
            if (positionSelection.x==100) positionSelection.x=650;
        break;
        case SDLK_LEFT:
            if (positionSelection.x==650) positionSelection.x=100;
        break;
        default:
        break;
    }
break;

case SDL_MOUSEMOTION : //souris
SDL_GetMouseState( &xmouse, &ymouse );
if((ymouse<200)&&(ymouse>50)) positionSelection.y=150;
if((ymouse<300)&&(ymouse>200)) positionSelection.y=250;
if((ymouse<400)&&(ymouse>300)) positionSelection.y=350;
if((ymouse<500)&&(ymouse>400)) positionSelection.y=450;
if((ymouse<600)&&(ymouse>500)) positionSelection.y=550;
if(xmouse<400) positionSelection.x=100;
if(xmouse>400) positionSelection.x=650;
break;
case SDL_MOUSEBUTTONDOWN:
continuer = 0;
break;
```

```
}



/* On met à jour l'affichage */
SDL_Flip(ecran);

    }


////////////////////////////////////////
/*Choix morceau*/
if ((positionSelection.y==150)&&(positionSelection.x==100)) choix=10;
if ((positionSelection.y==250)&&(positionSelection.x==100)) choix=11;
if ((positionSelection.y==350)&&(positionSelection.x==100)) choix=12;
if ((positionSelection.y==450)&&(positionSelection.x==100)) choix=13;
if ((positionSelection.y==550)&&(positionSelection.x==100)) choix=14;
if ((positionSelection.y==150)&&(positionSelection.x==650)) choix=15;
if ((positionSelection.y==250)&&(positionSelection.x==650)) choix=16;
if ((positionSelection.y==350)&&(positionSelection.x==650)) choix=17;
if ((positionSelection.y==450)&&(positionSelection.x==650)) choix=18;
if ((positionSelection.y==550)&&(positionSelection.x==650)) choix=19;



//Libération de l'espace
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Selection);
return choix; //On retourne la valeur du morceau choisi
}



/*************************************************************************
*************************************************************************
**********

                                FONCTION CLAVIER/ARDUINO

        ------------------------------------------------------------------

    On donne le choix entre jouer avec le clavier ou avec Arduino
    (on peut jouer avec le clavier quand on lance arduino)

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main
    RETURN => 0 si on choisit le clavier et 1 si on choisit Arduino


                                                                    Auteur
: Jean

*************************************************************************
*************************************************************************
**********/

int clavierArduino(SDL_Surface* ecran)
{
SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *selectArduino = NULL, *selectClavier;
```

```
SDL_Rect positionFond,positionSelection1,positionSelection2;
positionFond.x = 0; positionFond.y = 0;
positionSelection1.x = 150; positionSelection1.y = 400; //Clavier
positionSelection2.x = 800; positionSelection2.y = 400; //Arduino
imageDeFond = SDL_LoadBMP("images/clavierArduino.bmp"); selectArduino =
SDL_LoadBMP("images/selectArduino.bmp"); selectClavier =
SDL_LoadBMP("images/selectClavier.bmp");//images
SDL_SetColorKey(selectClavier, SDL_SRCCOLORKEY, SDL_MapRGB(selectClavier-
>format, 0,0, 255));//transparence
SDL_SetColorKey(selectArduino, SDL_SRCCOLORKEY, SDL_MapRGB(selectArduino-
>format, 0,0, 255));//transparence

int continuer = 1,xmouse=0,ymouse=0,choix = 0;


while (continuer)
    {


SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir

SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

if (choix)  SDL_BlitSurface(selectArduino, NULL, ecran,
&positionSelection2);
else  SDL_BlitSurface(selectClavier, NULL, ecran, &positionSelection1);

SDL_WaitEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: // clavier

    switch(event.key.keysym.sym)
    {
        case SDLK_RETURN:
        continuer=0;
        break;
        case SDLK_RIGHT:
        choix = 1;
        break;
        case SDLK_LEFT:
        choix = 0;
        break;
        default:
        break;
    }
break;

case SDL_MOUSEMOTION : //souris
SDL_GetMouseState( &xmouse, &ymouse );
if(xmouse<600) choix = 0;
else choix = 1;
break;
case SDL_MOUSEBUTTONDOWN:
continuer = 0;
break;
```

```
}



 //On met à jour l'affichage
SDL_Flip(ecran);

    }


//////////////////////////////////////////


//Libération de l'espace
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(selectClavier);
SDL_FreeSurface(selectArduino);
return choix; //On retourne la valeur du morceau choisi



}

/*************************************************************************
*************************************************************************
**********

                                FONCTION DIFFICULTE

        ------------------------------------------------------------------

    On donne le choix entre jouer en mode facile, normal ou difficile
                        (1/3 des notes,1/2 des notes, toutes les notes)

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main
    RETURN => 0 pour facile, 1 pour normal et 2 pour difficile


                                                                Auteur
: Jean

*************************************************************************
*************************************************************************
**********/
int difficult(SDL_Surface* ecran)
{
SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *Selection = NULL;
SDL_Rect positionFond,positionSelection;
positionFond.x = 0;
positionFond.y = 0;
positionSelection.x = 370;
positionSelection.y = 400;
imageDeFond = SDL_LoadBMP("images/difficult.bmp"); Selection =
SDL_LoadBMP("images/selection.bmp");//on indique ou est l'image de fond
SDL_SetColorKey(Selection, SDL_SRCCOLORKEY, SDL_MapRGB(Selection->format,
0,0, 255));//transparence
```

```
int continuer = 1,xmouse=0,ymouse=0,difficulte = 1;



while (continuer)
    {

            /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On remet le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);
SDL_BlitSurface(Selection, NULL, ecran, &positionSelection);



SDL_WaitEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: // clavier

    switch(event.key.keysym.sym)
    {
        case SDLK_RETURN:
        continuer=0;
        break;
        case SDLK_UP:
            if (positionSelection.y > 300)
positionSelection.y=positionSelection.y-170;
        break;
        case SDLK_DOWN:
            if (positionSelection.y < 500)
positionSelection.y=positionSelection.y+170;
        break;
        default:
        break;
    }
break;

case SDL_MOUSEMOTION : //souris
SDL_GetMouseState( &xmouse, &ymouse );
if((ymouse<300)&&(ymouse>50)) positionSelection.y=230;
if((ymouse<450)&&(ymouse>300)) positionSelection.y=400;
if((ymouse<600)&&(ymouse>450)) positionSelection.y=570;
break;
case SDL_MOUSEBUTTONDOWN:
continuer = 0;
break;

}

if (positionSelection.y==230) positionSelection.x = 400;
else positionSelection.x = 370;


/* On met à jour l'affichage */
SDL_Flip(ecran);
```

```
    }

////////////////////////////////////////
/*Choix difficulté*/
if (positionSelection.y==230) difficulte=0;
if (positionSelection.y==400) difficulte=1;
if (positionSelection.y==570) difficulte=2;


//Libération de l'espace
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Selection);
return difficulte; //On retourne la valeur du morceau choisi



}
```

Tuto.c

```
#include <stdlib.h>
#include <stdio.h>
#include <SDL/SDL.h>
#include <SDL/SDL_ttf.h>
#include <fmod.h>

#define TAILLE_MAX 10000 // Tableau de taille 10 000

void demo (SDL_Surface* ecran);//Comme il n'y qu'un seul prototype faire un
.h semble inutile

/*************************************************************************
*************************************************************************
**********

                                FONCTION TUTO

        ------------------------------------------------------------------

    On affiche l'image qui permet d'expliquer comment jouer et on lance une
musique on propose une démo si on appuie sur espace

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main


                                                                    Auteur
: Aurele

*************************************************************************
*************************************************************************
**********/



void tuto(SDL_Surface* ecran)
```

```
{


SDL_ShowCursor(SDL_DISABLE);//On n'affiche plus le curseur
SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *Noms = NULL;
SDL_Rect positionFond,positionNoms;
int etat = 0;

positionFond.x = 0;
positionFond.y = 0;
positionNoms.x = 400;
positionNoms.y = 900;

imageDeFond = SDL_LoadBMP("images/tuto.bmp");//on indique ou est l'image de
fond

    /*Musique*/
FMOD_SYSTEM *system;
FMOD_SOUND *musique;
FMOD_RESULT resultat;
FMOD_System_Create(&system);
FMOD_System_Init(system, 1, FMOD_INIT_NORMAL, NULL);
/* On ouvre la musique*/
resultat = FMOD_System_CreateSound(system, "musique.mp3",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);

/* On joue la musique*/
FMOD_System_PlaySound(system, FMOD_CHANNEL_FREE, musique, 0,NULL);




            /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On met le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);



int continuer = 1;
while (continuer)
    {



    SDL_PollEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: /* Si appui sur une touche */

    switch(event.key.keysym.sym)
    {
        case SDLK_ESCAPE:
        continuer=0;
        break;
        case SDLK_SPACE:
```

```
        FMOD_Sound_Release(musique);
        demo(ecran);
        etat = 1;//la musique a déjà été arretée, on ne l'arretera pas deux
fois
        continuer=0;
        break;
        default:
        break;
    }
break;

}



/* On met à jour l'affichage */
SDL_Flip(ecran);

    }


////////////////////////////////////////
/*Sortie de boucle*/
SDL_ShowCursor(SDL_ENABLE);//on réaffiche le curseur pour le menu
//Libération de l'espace
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Noms);
//pour la musique

if(etat==0)FMOD_Sound_Release(musique);
FMOD_System_Close(system);
FMOD_System_Release(system);

}



/*************************************************************************
**************************************************************************
**********

                                FONCTION DEMO

        ------------------------------------------------------------------

   On a copié la fonction jeu en faisant faire un perfect sur frère jacques
par
   l'ordinateur pour faire la démo

        ------------------------------------------------------------------
    ecran : fenêtre initialisée dans le main


                                                                    Auteur
: Aurele

*************************************************************************
**************************************************************************
**********/
```

```
void demo (SDL_Surface* ecran)
{
    SDL_Event event; //pour pouvoir gerer les events
SDL_Surface *imageDeFond = NULL, *Note = NULL, *Note_do = NULL, *Note_re =
NULL, *Note_mi = NULL, *Note_fa = NULL, *Note_sol = NULL, *Note_la = NULL,
*Note_si = NULL; //Initialisation des images : on crée un pointeur pour
chaque image auquel on met la valeur NULL
SDL_Rect positionFond,positionNote[10000],
positionNote_do,positionNote_re,positionNote_mi,positionNote_fa,positionNot
e_sol,positionNote_la,positionNote_si; //Initialisation des positions des
images

//Initialisation positions x et y
int i; for(i=0;i<10000;i++)
        {positionNote[i].x=0; positionNote[i].y=0;} //Initialisation de
TOUTES les notes du morceau

positionNote_do.y = positionNote_re.y = positionNote_mi.y =
positionNote_fa.y = positionNote_sol.y = positionNote_la.y =
positionNote_si.y = 658;
positionNote_do.x = 265;
positionNote_re.x = 408;
positionNote_mi.x = 548;
positionNote_fa.x = 690;
positionNote_sol.x = 834;
positionNote_la.x = 973;
positionNote_si.x = 1117;
positionFond.x = 0;
positionFond.y = 0;


imageDeFond = SDL_LoadBMP("images/fond.bmp");//on indique ou est l'image de
fond


//Images
Note = SDL_LoadBMP("images/notes/note.bmp") ;
Note_do = SDL_LoadBMP("images/notes/do.bmp");
Note_re = SDL_LoadBMP("images/notes/re.bmp");
Note_mi = SDL_LoadBMP("images/notes/mi.bmp");
Note_fa = SDL_LoadBMP("images/notes/fa.bmp");
Note_sol = SDL_LoadBMP("images/notes/sol.bmp");
Note_la = SDL_LoadBMP("images/notes/la.bmp");
Note_si = SDL_LoadBMP("images/notes/si.bmp");
SDL_SetColorKey(Note, SDL_SRCCOLORKEY, SDL_MapRGB(Note->format, 0,0,
255));//transparence
//

 FILE* fichier = NULL;//pour lire la "partition"



/***********Musique***************
*******************************/

FMOD_SYSTEM *system;
FMOD_SOUND *musique;//musique :/
FMOD_RESULT resultat;
FMOD_System_Create(&system);
FMOD_System_Init(system, 1, FMOD_INIT_NORMAL, NULL);
```

```
    resultat = FMOD_System_CreateSound(system, "musiques/FrereJacques.mid",
FMOD_SOFTWARE | FMOD_2D | FMOD_CREATESTREAM, 0, &musique);
    fichier = fopen("musiques/FrereJacques.txt", "r"); //Le fichier texte
qui contient la "partition"



   if (resultat != FMOD_OK)//verification que la musique marche
    {
        fprintf(stderr, "Impossible de lire pan.wav\n");
        exit(EXIT_FAILURE);
    }


 /*****************************
 Initialisation pour le texte
 *****************************/
    char caracteres[20] = ""; // Tableau de char suffisamment grand pour le
score
    TTF_Init(); //Initialisation de la banque de donnée pour le texte
    SDL_Surface *score = NULL;
    SDL_Rect position;
    TTF_Font *police = NULL; //TTF_OpenFont doit stocker son résultat dans
une variable de type TTF_Font
    SDL_Color couleurNoire = {0, 0, 0}; //couleur police => noir
    police = TTF_OpenFont("police.ttf", 70);//police choisie, taille police
////////////////////////////////////////////////

 /************************************************
       +Lecture du fichier texte (Yacine)
       +On met les notes a leur place (do,ré...)
           en fonction de la difficulté
 ************************************************/


char chaine[TAILLE_MAX]="";
int debut[TAILLE_MAX ];
int note[TAILLE_MAX ];
int tempsFin = 0 ;
 int j=0;
 while (fgets(chaine, TAILLE_MAX, fichier) != NULL) //tant que le fichier
n'a pas été totalement parcouru (fgets s'incremente automatiquement)
   {

           //On prend toutes les notes
                   sscanf(chaine, "%d - %d", &debut[j], &note[j]); //
recupere la note et la date

switch (note[j])//On met en place les notes
{

   case 0 : //do
        positionNote[j].x = 250;
        break;
   case 1 : //ré
        positionNote[j].x = 395;
        break;
   case 2 : //mi
        positionNote[j].x = 525;
        break;
```

```
   case 3 : //fa
        positionNote[j].x = 680;
        break;
   case 4 : //sol
        positionNote[j].x = 820;
        break;
   case 5 : //la
        positionNote[j].x = 960;
        break;
   case 6 : //si
        positionNote[j].x = 1100;
        break;
    case 8 : // Pour la fin
        tempsFin = debut[j];
        positionNote[j].x = 20000;//pour pas afficher la note
   default :
        break;

        }
        j++;



   }


 ////////////////////////////////////////////
int k=0;//notes 1,2,3....
int tempsDebut=SDL_GetTicks();//SDL_GetTicks donne le temps qu'il s'est
écoulé depuis le lancement du programme, on retire donc le temps qu'il
s'est écoulé entre le lancement et le début du morceau
int tempsPrecedent = 0, tempsActuel = 0; //Timer (temps note permet de
savoir a quel instant t la note a été jouée
int continuer = 1;

 //Boucle jeu (ici démo!)
while (continuer)
{


/* On joue la musique au bon moment de manière à ce qu'elle soit
synchronisée avec les notes qui défilent */
if ((tempsActuel>=2700)&&(tempsActuel<=2750))FMOD_System_PlaySound(system,
FMOD_CHANNEL_FREE, musique, 0,NULL);


    /* On efface l'écran */
SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 255,255,
255));//255,255, 255 veut dire qu'on met un ecran noir
    /* On remet le fond */
SDL_BlitSurface(imageDeFond, NULL, ecran, &positionFond);

tempsActuel = SDL_GetTicks() - tempsDebut; //temps



/*********************************************
 + On affiche les notes
 + On fait descendre les notes le long des lignes
    J'ai galeré pour ca !
```

```
**********************************************/

    if(tempsActuel>=debut[k+1]) k++; // passage à la note suivante


int l=k;

    do
    {
            positionNote[l].y= positionNote[l].y+tempsActuel/10*2-
tempsPrecedent/10*2;//descente de la note en utilisant le temps comme
réference (elle met du coup 2.7sec a desendre)

            if (positionNote[l].y>575) positionNote[l].y=10000; // si la
note arrive en bas on la fait "disparaitre"

        SDL_BlitSurface(Note, NULL, ecran, &positionNote[l]);//on affiche
les notes
        l--;
    }while(l>=0);



///La boucle do while est la car il faut la faire au moins une fois quand
il n'y a qu'un seule note
///Pour afficher les notes précédentes (sinon on a qu'une seule note
affichée)


/*Pour sortir du morceau à la fin*/
if (tempsActuel >= tempsFin) continuer = 0;



///


tempsPrecedent = tempsActuel; // comme on utilise le temps pour la boucle
d'avant on le met ici (on pourrait le mettre à la toute fin, ce qui serait
plus logique mais ici on comprend mieux)




SDL_PollEvent(&event);
switch(event.type)
{
case SDL_QUIT:
exit(EXIT_FAILURE);
break;
case SDL_KEYDOWN: /* Si appui sur une touche */

    switch(event.key.keysym.sym)
    {

        case SDLK_ESCAPE:
        continuer=0;
        break;
```

```
        default:
        break;
    }
break;

}

l=k;//on teste toutes les notes qui on étées affichées

    do
    {
        //démonstration : on fait un perfect
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
250))  SDL_BlitSurface(Note_do, NULL, ecran, &positionNote_do);
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
395))  SDL_BlitSurface(Note_re, NULL, ecran, &positionNote_re);
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
525))  SDL_BlitSurface(Note_mi, NULL, ecran, &positionNote_mi);
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
680))  SDL_BlitSurface(Note_fa, NULL, ecran, &positionNote_fa);
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
820))  SDL_BlitSurface(Note_sol, NULL, ecran, &positionNote_sol);
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
960))  SDL_BlitSurface(Note_la, NULL, ecran, &positionNote_la);
        if
((positionNote[l].y>550)&&(positionNote[l].y<580)&&(positionNote[l].x ==
1100)) SDL_BlitSurface(Note_si, NULL, ecran, &positionNote_si);

        l--;
    }while(l>=0); //j'aime bien les do while (mêmes raisons qu'au dessus)

/***********************************
Fonction pour le texte, ici le score
et le nom du morceau
***********************************/

//En cas d'erreur (plus propre)
if(TTF_Init() == -1)
{

    fprintf(stderr, "Erreur d'initialisation de TTF_Init : %s\n",
TTF_GetError());
    exit(EXIT_FAILURE);
}


/*titre*/


       sprintf(caracteres, "Frère Jacques");


     SDL_FreeSurface(score);//On efface la surface précédente (sinon ca
prend 2go de RAM)
     score = TTF_RenderText_Solid(police, caracteres, couleurNoire);
```

```
    //Position score//
        position.x = 20;
        position.y = 60;
        SDL_BlitSurface(score, NULL, ecran, &position); /* Blit du texte*/


////////////////////////////////////////////////////////////////////////////
//////////////

/* On met à jour l'affichage */
SDL_Flip(ecran);

}

/////////////////////////////////////////
/*Sortie de boucle*/
/////////////////////////////////////////
    //images
SDL_FreeSurface(imageDeFond);
SDL_FreeSurface(Note);
SDL_FreeSurface(Note_do);
SDL_FreeSurface(Note_re);
SDL_FreeSurface(Note_mi);
SDL_FreeSurface(Note_fa);
SDL_FreeSurface(Note_sol);
SDL_FreeSurface(Note_la);
SDL_FreeSurface(Note_si);
    //pour la musique
FMOD_Sound_Release(musique);
FMOD_System_Close(system);
FMOD_System_Release(system);

}
```

Arduino - Piano Hero

Arduino - Piano Hero.ino

```
int buttonState = 0;         // ces variables serviront a stocker l'etat de
chaque bouton (ennfoncé ou pas)
int buttonState2 = 0;
int buttonState3 =  0;
int buttonState4 =  0;
int buttonState5 =  0;
int buttonState6 =  0;
int buttonState7 =  0;

void setup() {

  // initialise les pin ou les boutons sont branchées comme des entrées.
  pinMode(2, INPUT);
```

```
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
  pinMode(8, INPUT);


Serial.begin(9600); // initialise la communication série a 9600 bauds.


}

void loop(){ // cette boucle se répète a l'infini tant que la carte arduino
est alimentée.

  // on lit l'etat de chaque bouton et on le stocke dans la variable prevue
a cet effet.
 buttonState = digitalRead(2);
 buttonState2 = digitalRead(3);
 buttonState3 = digitalRead(4);
 buttonState4 = digitalRead(5);
 buttonState5 = digitalRead(6);
  buttonState6 = digitalRead(7);
 buttonState7 = digitalRead(8);



  if (buttonState == HIGH) // si le bouton est enfoncé
  {

    Serial.print("1");  // on ecrit un caractère propre a ce bouton dans le
port serie

  }

    if (buttonState2 == HIGH)
    {

    Serial.print("2");
  }

     if (buttonState3 == HIGH)
    {

    Serial.print("3");
  }
  else

    if (buttonState4 == HIGH)
    {

    Serial.print("4");
  }

   if (buttonState5 == HIGH)
   {

   Serial.print("5");
  }
```

```
    if (buttonState6 == HIGH)
    {

   Serial.print("6");
  }

    if (buttonState7 == HIGH)
    {

   Serial.print("7");
  }


  Serial.println("\0"); //on ecrit ce caractere pour que l'ordinateur
comprenne qu'il s'agit de la fin d'une chaine de caractères.


}
```

Créateur de partitions

Créateur de partitions.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>

#define TAILLE_MAX 1000 // Tableau de taille 1000

/* prototypes des fonctions:  */
int chercherInstrument (char *chaine); //determine quels instruments sont
presents dans le fichier midi
int analyserNote(chaine);//recupere la frequence de chaque note
int recupererDivision(char *chaine); //recupere le nombre de ticks par
temps (le tick est une unité de temps propre aux fichies midi)
int recupererSignature(char *chaine);
int recupererBPM(char *chaine);
int determinerDebut(char *chaine,int bpm,int division,int nbtemps,int*
Pmesureprecedente, int* Ptempsprecedent, int* Ptickprecedent); //determine
le temps ecoulé avant le debut d'une note



int main(int argc, char *argv[])
{


   int position[128]; //pour chaque instrument: position de la track dans
le fichier texte (position du curseur)
   int frequence[5000]; //la frequence de chacune des notes
```

```
   int debut[5000]; //la durée en millisecondes avant la debut de la note
   int note[5000]; //chaque note prend au final une valeur de 0 a 6 (0=do,
1=ré...)
   int i=0, j=0, k=0; //variables permettant de parcourir des tableaux
   int pgm, choix; // le pgm est un numero propre a chaque instrument dans
les fichiers midi
   int bpm, nbtemps, division; //battement par minute, nombre de temps par
mesure, et nombre de divisions par temps.
   int mesureprecedente=1, tempsprecedent=1, tickprecedent=0;


FILE* fichier = NULL;
char chaine[TAILLE_MAX] = "";
fichier = fopen("morceau.txt", "r"); //le fichier texte contenant toutes
les informations a recuperer (notes..)

FILE* fichier2 = NULL;
fichier2 = fopen("test.txt", "w");  //fichier texte de sortie
char chainesortie[]="";  //la chaine qui sera ecrite dans le fichier texte
de sortie.

if (fichier != NULL) //si le fichier a bien été ouvert
{
    char *chaine1;

chaine1=fgets(chaine, TAILLE_MAX, fichier); //recupere la premiere ligne du
fichier texte
division=recupererDivision(chaine1); //recupere le nombre de ticks par
temps (le tick est une unité de temps propre aux fichiers midi), cette
information se trouve toujours a la premiere ligne du fichier texte

printf("Quelle piste voulez vous recuperer ?:\n\n");

while (fgets(chaine, TAILLE_MAX, fichier) != NULL) //tant que le fichier
n'a pas été totalement parcouru (fgets s'incremente automatiquement)
    {
             if(strstr(chaine, "pgm #=  ")!=NULL   || strstr(chaine, "pgm
#= ")!=NULL)// si un numero d'instrument est present sur la ligne
             {
                 i=chercherInstrument(chaine);
                 position[i]=ftell(fichier); // On retient la position du
curseur au debut de la piste
              }
              else if(strstr(chaine, "Time Sig")!=NULL)//ou si cette ligne
donne la signature rythmique du morceau
              {
                  nbtemps=recupererSignature(chaine);
              }
              else if(strstr(chaine, "Tempo")!=NULL)//ou si cette ligne
donne la BPM du morceau
              {
                 bpm=recupererBPM(chaine);
              }
    }

scanf("%d", &choix); //choix de la piste a analyser
fseek(fichier,position[choix],SEEK_SET); //on positionne le "curseur" a la
ligne suivant le pgm
```

```
while( fgets(chaine, TAILLE_MAX, fichier)!= NULL   && strstr(chaine, "End
of track")==NULL)
    {
        determinerDebut(chaine, bpm, division, nbtemps, &mesureprecedente,
&tempsprecedent, &tickprecedent);

         if(strstr(chaine, "On Note")!=NULL) // On repere si la ligne
correspond au debut d'une note
             {

                 frequence[j]=analyserNote(chaine); //on determine la
frequence de chaque note
                 debut[j]=determinerDebut(chaine, bpm, division, nbtemps,
&mesureprecedente, &tempsprecedent, &tickprecedent);

                 if (debut[j]==0) //si determinerDebut renvoie 0 cela veut
dire que la ligne analysée ne donne aucune indication concernant le temps,
celui ci correspond donc au temps de la note precedente
                 {
                    debut[j]=debut[j-1];
                 }
                 j++;
             }
    }


for(k=0;k<j ;k++)
{
     switch(frequence[k])   // ce switch permet d'associer a chaque
frequence une valeur entre 0 et 6 (0= do, 1= ré ...)
    {
        case 32:
           printf("  %dms", debut[k]);
           note[k]=0;
        break;
         case 37:
            printf("  %dms", debut[k]);
            note[k]=1;
        break;
         case 41:
             printf("  %dms", debut[k]);
             note[k]=2;
        break;
         case 44:
            printf("  %dms", debut[k]);
             note[k]=3;
        break;
         case 49:
             printf("  %dms", debut[k]);
              note[k]=4;
        break;
         case 55:
               printf("  %dms", debut[k]);
               note[k]=5;
        break;
         case 61:
                printf("  %dms", debut[k]);
                note[k]=6;
        break;
    }
printf("\n");
```

```
sprintf(chainesortie, "%d - %d \n",debut[k], note[k]); //on ecrit le date
de debut et la valeur entre 0 et 6 de chaque note dans une chaine


 fputs(chainesortie, fichier2); //on place cette chaine dans le fichie
texte de sortie
}



/*Lorsque la note vaut 8 le programme de jeu comprend qu'il s'agit de la
fin du morceau.
en fonction du niveau de difficulté, le programme de jeu peut sauter
certaines notes c'est pourquoi j'ecris la note de fin 4 fois*/

for(j=0; j<5; j++)
{
    sprintf(chainesortie, "%d - 8\n",debut[k-1]+3000);
   //je reecris la note precedente en rajoutant 3 secondes pour que le jeu
ne s'interrompe pas tout de suite apres la derniere note.
    fputs(chainesortie, fichier2);
}







fclose(fichier);
fclose(fichier2);
}
return 0;
}



/**************************************************************************
************************************************************
                                FONCTION CHERCHERINSTRUMENT
        ------------------------------------------------------------------
Cette fonction permet de determiner quels instruments sont présents dans le
morceau, d'afficher leurs noms afin de laisser a l'utilisateur
lec choix de la piste qu'il souhaite récuperer.
Elle prend comme paramètre une chaine de caractere contenant la ligne
actuelle du fichier texte parcouru.
                                                                    Auteur
: Yacine Saoudi

**************************************************************************
************************************************************/
int chercherInstrument (char *chaine)
{
 int pgm;

char *chaine1;
char *chaine2;
```

```
 chaine1=strtok(chaine, "#"); //on coupe la chaine autour du signe #
 chaine2=strtok(NULL, "#"); // on recupere la partie droite de la chaine
precedente
 sscanf(chaine2, "=  %d", &pgm); // recupere le numero de l'instrument


switch(pgm) //le  standard midi associe a chaque numero de pgm un
instrument, seuls quelques uns sont gérés par cette fonction:
 {
     case 0:
     printf("%d-Acoustic Grand Piano \n",pgm);
     break;

  case 1:
     printf("%d-Bright Acoustic Piano \n",pgm);
     break;

  case 2:
     printf("%d-Electric grand Piano \n",pgm);
     break;

  case 3:
     printf("%d-Honky Tonk Piano \n",pgm);
     break;

  case 4:
     printf("%d-Electric Piano 1 \n",pgm);
     break;

  case 5:
     printf("%d-Electric Piano 2 \n",pgm);
     break;

  case 6:
     printf("%d-Harpsichord \n",pgm);
     break;

  case 7:
     printf("%d-Clavinet \n",pgm);
     break;
  case 24:
     printf("%d-Nylon Accoustic Guitar \n", pgm);
     break;

  case 25:
     printf("%d-Steel Acoustic Guitar \n", pgm);
     break;

  case 26:
     printf("%d-Jazz Electric Guitar \n", pgm);
     break;

  case 27:
     printf("%d-Cieán Electric Guitar \n", pgm);
     break;

  case 28:
     printf("%d-Muted Electric Guitar \n", pgm);
     break;
```

```
  case 29:
     printf("%d-Overdrive Guitar \n", pgm);
     break;

  case 30:
     printf("%d-Distorted Guitar \n", pgm);
     break;

  case 31:
     printf("%d-Guitar Harmonics \n", pgm);
     break;
  default:
     break;
 }

return pgm;
}



/**************************************************************************
*************************************************************
                                FONCTION ANALYSERNOTE
        ------------------------------------------------------------------
Cette fonction permet de determiner la frequence de chaque note.
Elle prend comme paramètre une chaine de caractere contenant la ligne
actuelle du fichier texte parcouru.


Remarque: nous pourrions directement renvoyer une valeur entre 0 et 6
(0=do, ré=1, mi=2...),
cela eviterait le long switch a la fin du programme.
 Cependant nous avons préféré laisser cette fonction telle quelle au cas ou
nous desirions adapter la manette pour gerer pus d'un octave
 ou creer une manette guitare.


                                                                    Auteur
: Yacine Saoudi

**************************************************************************
*************************************************************/
int analyserNote(chaine)
{
    int frequence, octave;
    char note;
    char *chaine1;
    char *chaine2;


    chaine1=strtok(chaine, "i"); //permet de decouper la chaine en 2 autour
du i car c'est le seul caractere qui n'apparait qu'avant la partie a
recuperer
    chaine2=strtok(NULL, "i"); //on recupere la chaine a droite du
caractere i
    chaine1=strtok(chaine2, "|"); //on tronque ensuite toute la partie
concernant le volume
    chaine2=strtok(chaine1, "="); //on coupe autour du =
    chaine1=strtok(NULL, "="); //on recupere la partie a droite du =
```

```
   if(strstr(chaine1, "#")==NULL) // S'il n y a pas de # dans la chaine
              {
    sscanf(chaine1, "%c %d", &note, &octave); // recupere note et octave

              }
    else if(strstr(chaine1, "#")!=NULL) // S'il y a un # dans la chaine
(do#, ré#...)
    {
        sscanf(chaine1, "%c#%d", &note, &octave); // recupere note et
octave
    }


    switch(note) //les fichiers midi utilisent la notation anglo-saxonne
(les notes en minuscules ou majuscules sont exactement les memes)
    {
        case 'c':
        frequence=32;  //frequence du DO 0, les frequences sont
approximatives et ne prennent pas en compte les dièses et bémol
        break;
         case 'd':
        frequence=37; // ré
        break;
         case 'e':
        frequence=41; // mi
        break;
         case 'f':
        frequence=44;
        break;
         case 'g':
        frequence=49;
        break;
         case 'a':
        frequence=55;
        break;
         case 'b':
        frequence=61;
        break;
          case 'C':
        frequence=32;  //frequence du DO 0
        break;
         case 'D':
        frequence=37;
        break;
         case 'E':
        frequence=41;
        break;
         case 'F':
        frequence=44;
        break;
         case 'G':
        frequence=49;
        break;
         case 'A':
        frequence=55;
        break;
         case 'B':
        frequence=61;
        break;
    }
```

```
/*
if(octave>0)
{
    frequence=frequence*2*octave; //pour passer d'un octave a l'autre i
faut multipler la frequence par 2
}
*/
    return frequence;
}


/**************************************************************************
************************************************************
                                FONCTION RECUPERERDIVISION
        ------------------------------------------------------------------
Cette fonction permet de determiner le nombre de divisions par temps (afin
de calculer la durée d'un tick par la suite).
Elle prend comme paramètre une chaine de caractere contenant la ligne
actuelle du fichier texte parcouru.

                                                                    Auteur
: Yacine Saoudi
**************************************************************************
************************************************************/
int recupererDivision(char *chaine) //recupere le nombre de ticks par temps
(le tick est une unité de temps propre aux fichies midi)
{
    int tick, division;
    char *chaine1, *chaine2;

     chaine1=strtok(chaine, "n"); //decoupe la chaine autour du 'n' de
"division"
    chaine2=strtok(NULL, "n"); //on recupere la chaine a droite du
caractere n
    sscanf(chaine2, "=%d", &division);
    return division;
}


/**************************************************************************
************************************************************
                                FONCTION RECUPERERSIGNATURE
        ------------------------------------------------------------------
Cette fonction permet de determiner le nombre de temps par mesure (afin de
calculer la durée d'une mesure par la suite).
Elle prend comme paramètre une chaine de caractere contenant la ligne
actuelle du fichier texte parcouru.

                                                                    Auteur
: Yacine Saoudi
**************************************************************************
************************************************************/
int recupererSignature(char *chaine)
{
int nbtemps;
char *chaine1, *chaine2;

    chaine1=strtok(chaine, "g"); //On coupe la chaine apres le "Time sig"
    chaine2=strtok(NULL, "g"); //on recupere la partie a droite du
caractere "g"
```

```
    chaine1=strtok(chaine2, "|"); //On coupe la chaine apres le "|"
    chaine2=strtok(NULL, "|"); //on recupere la partie a droite du
caractere "|"

    sscanf(chaine2, "%d/", &nbtemps); // recupere le nombre de temps par
mesure

   return nbtemps;
}


/**************************************************************************
************************************************************
                              FONCTION RECUPERERBPM
        ------------------------------------------------------------------
Cette fonction permet de determiner le nombre de battements par minute.
Elle prend comme paramètre une chaine de caractere contenant la ligne
actuelle du fichier texte parcouru.


Remarque: de nombreux fichiers midi changent de BPM en cours de chanson,
adapter ce programme pour qu'il prenne en compte ces variations
permettrait d'assurer une compatibilité avec un nombre beaucoup plus grand
de fichiers midi.

                                                                    Auteur
: Yacine Saoudi
**************************************************************************
************************************************************/
int recupererBPM(char *chaine)
{
    int bpm;
char *chaine1, *chaine2;

    chaine1=strtok(chaine, "="); //On coupe la chaine autour du "="
    chaine2=strtok(NULL, "="); //on recupere la partie a droite
    chaine1=strtok(chaine2, "|"); //On coupe la chaine autour du "|"

    sscanf(chaine1,"%d", &bpm);

    return bpm;
}

/**************************************************************************
************************************************************
                              FONCTION DETERMINERDEBUT
        ------------------------------------------------------------------
Cette fonction permet de determiner la date ou une note est jouée sur le
morceau.
Elle prend comme paramètre une chaine de caractere contenant la ligne
actuelle du fichier texte parcouru, toutes les informations relatives
au tempo de la musique (bpm, nombre de divisions par temps, nombre de temps
par mesure).

Si la ligne analysée ne donne aucune (ou seulement certaines) indication
sur le temps, on reprend alors les valeurs de la derniere note.
C'est pourquoi cette fonction prend egalement en parametres la date de la
derniere mesure, du dernier temps et du dernier tick.
                                                                    Auteur
: Yacine Saoudi
```

```
**************************************************************************
************************************************************/
int determinerDebut(char *chaine,int bpm,int division,int nbtemps,int*
Pmesureprecedente, int* Ptempsprecedent, int* Ptickprecedent) //determine
le temps ecoulé avant le debut d'une note
{
   char chaine1[6] = {0};
char chaine2[9]= {0};

int mesure=0, temps=0, tick=0;

int dureetemps; //duree d'un temps (depend du bpm)
int dureemesure; //duree d'une mesure (depend du nombre de temps par mesure
et duree d'un temps)
int dureetick; //depend du nombre de ticks par temps (division)
int debut; //temps total ecoulé avant le debut de la note


strncpy(chaine1, chaine, 5); //on copie les 5 premiers caracteres de chaine
dans chaine1
strncpy(chaine2, chaine, 8); //on copie les 5 premiers caracteres de chaine
dans chaine2



if(strstr(chaine1, ":")!=NULL) // s'il y a un ':' dans les 5 premiers
caracteres de chaine cela veut dire que le temps est sous la forme  mesure:
temps: tick
{
    sscanf(chaine, "%d:%d:%d", &mesure, &temps, &tick); //on recupere les 3
valeurs
}
else if(strstr(chaine1, ":")==NULL && strstr(chaine2, ":")!=NULL)// s'il y
a un ':' dans les 8 premiers caracteres de chaine cela veut dire que le
temps est sous la forme  temps: tick
{
    sscanf(chaine, "%d:%d", &temps, &tick); //on recupere les 2 valeurs
}
else if(strstr(chaine2, ":")==NULL ) // s'il n' y a aucun ':' dans les 8
premiers caracteres alors, seuls les ticks sont indiqués
{
    sscanf(chaine, "%d", &tick);
}



if(mesure==0) //s'il n'y a aucune indication sur la mesure, cela veut dire
qu'il s'agit de la meme que la precedente.
{
    mesure= *Pmesureprecedente;
}
if(temps==0)
{
    temps= *Ptempsprecedent;
}
if(tick==0)
{
    tick= *Ptickprecedent;
}
```

```
//conversion en millisecondes:
dureetemps=60000/bpm;
dureemesure=dureetemps*nbtemps;
dureetick=dureetemps/division;

debut=(mesure-1)*dureemesure+(temps-1)*dureetemps+tick*dureetick;
// on doit toujours retirer 1 au temps et a la mesure car le tout debut du
morceau correspond au 1er temps de la 1ère mesure

*Pmesureprecedente=mesure;
*Ptempsprecedent=temps;
*Ptickprecedent=tick;
/*
Cette fonction ne peut renvoyer qu'une seule valeur, c'est pourquoi
j'uilise des pointeurs pour ces 3 variables car elles doivent etre
utilisées et modifiées a chaque fois que cette fonction s'execute.
*/

    return debut;
}
```