Programmation : Musique et dessin en langage C et SDL2 sur MorphOS

(Tutoriel développé et écrit par Sébastien Jeudy - mars 2025)

Ce tutoriel est la suite de celui-ci : obligement.free.fr/articles/animation_image_c_sdl2_morphos.php, dans lequel on avait introduit un cas simple d'utilisation de SDL2 en langage C : l'animation d'une image dans une fenêtre (un joli morpho bleu qui papillonne). Ceci sur MorphOS.

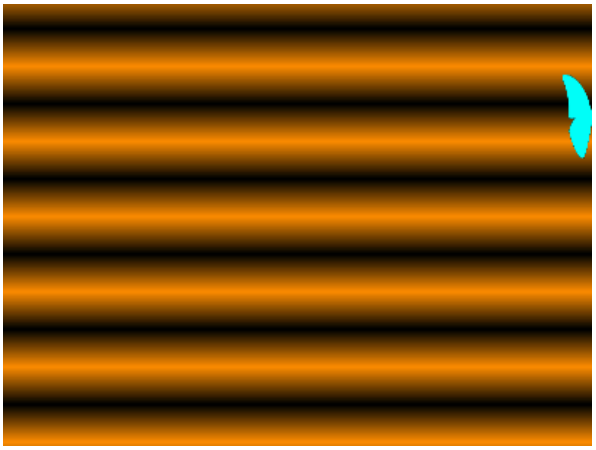
La bibliothèque SDL2 (bref rappel)

SDL2 (Simple DirectMedia Layer version 2) est une bibliothèque logicielle libre utilisée pour créer des applications multimédias en deux dimensions (jeux, démos,...). Son succès est dû à sa portabilité sur la plupart des plates-formes et à sa facilité d'implémentation dans de nombreux langages, dont le langage C (ce qui nous intéresse ici).

Une chance pour nous, SDL2 est également disponible sur MorphOS (et AmigaOS 4) depuis plusieurs années.

Présentation de l'évolution d'aujourd'hui

Dans cette deuxième version, on va ajouter à notre programme d'animation de la musique et des barres horizontales colorées qui défileront sous notre papillon (les fameuses "raster bars" en anglais).



Le résultat final
(image réduite, sans la musique ^^)

Toujours en langage C et sur MorphOS, mais le programme peut être recompilé sur d'autres plates-formes (en l'état).

La présentation est une nouvelle fois la plus détaillée possible afin d'être accessible au plus grand nombre avec seulement quelques rudiments en programmation. Il faut cependant avoir traité au préalable le précédent tutoriel.

Le code source complet du programme est disponible à la fin de l'article, avec des commentaires "// V2" pour repérer facilement les ajouts de cette deuxième version.

Prérequis (rappels)

Disposer de la dernière version de MorphOS : www.morphos-team.net/downloads.

Afin de disposer du langage C et de son compilateur GCC, télécharger et installer la dernière version du SDK de MorphOS (Software Development Kit ou trousse de développement logiciel) : www.morphos-team.net/downloads.

Puis télécharger et installer la dernière version de SDL2 sur MorphOS : www.morphos-storage.net/?find=SDL_2.

Fin de jeux (de P à Z)
Galerie de Mike Dafunk
Logos d'Obligement
Pubs pour matériels
Systèmes d'exploitation
Trombinoscope Alchimie 7
Vidéos

Téléchargement

Documents
Jeux
Logiciels
Magazines
Divers

Liens

Associations
Jeux
Logiciels
Matériel
Magazines et médias
Pages personnelles
Réparateurs
Revendeurs
Scène démo
Sites de téléchargement
Divers

Partenaires

A Propos

A propos d'Obligement

Contact

David Brunet

obligement@free.fr

Lors de l'installation, pour vos développements avec SDL2, attention à bien cocher l'option "Yes i have SDK installed" pour la question "Do you want SDK Files? (you need last SDK installed)".

Fichiers du programme étudié

- Code source : www.boingball.net/AMIGA_FOR_EVER/Codes/C/Butterfly2/Butterfly2.c.
- Programme exécutable (sur MorphOS) : www.boingball.net/AMIGA_FOR_EVER/Codes/C/Butterfly2/Butterfly2.
- Image bitmap (BMP) utilisée : www.boingball.net/AMIGA_FOR_EVER/Codes/C/Butterfly2/Butterfly.bmp.
- Musique utilisée (composée par notre ami Alexis "Ace" Mouth) : www.boingball.net/AMIGA_FOR_EVER/Codes/C/Butterfly2/SpaceLam3rs.mp3.

Tous ces fichiers doivent être dans le même répertoire.

Afin d'avoir tout de suite une idée du résultat, n'hésitez pas à lancer le programme exécutable "Butterfly2" (pour le quitter : touche "Échap" ou un clic sur la croix de la fenêtre).

Venons-en maintenant à l'explication détaillée des ajouts au code source, de la première ligne à la dernière ligne.

Attention : le langage C est sensible à la casse, il fait la différence entre les minuscules et les majuscules.

Inclusion du fichier d'en-tête "SDL_mixer.h" de la bibliothèque SDL2

```
#include <SDL2/SDL_mixer.h> // V2
```

Au début du code source (sous "SDL2/SDL.h"), ce fichier d'en-tête contient les déclarations et les définitions nécessaires pour utiliser les fonctionnalités audio de SDL2 dans notre programme (comme jouer la musique ajoutée).

Déclaration des variables globales à tout le programme

```
Mix_Music *musique; // V2
```

Suite aux autres déclarations, cette ligne déclare le pointeur "musique" vers un objet de type Mix_Music (utilisé pour charger notre musique à partir de son fichier).

Dans la définition de la fonction "fin"

Pour rappel, cette fonction est appelée pour quitter proprement le programme en libérant les ressources allouées.

```
if (musique) Mix_FreeMusic(musique); // V2
```

Sous "void fin()", cette ligne "if" teste si l'objet "musique" est défini et dans ce cas le détruit avec la fonction SDL2 Mix_FreeMusic.

```
Mix_CloseAudio(); // V2
```

Cette fonction ferme le système audio initialisé par SDL_mixer (via la fonction Mix_OpenAudio). Elle désactive et libère donc les ressources audio.

Dans la définition de la fonction "main"

Pour rappel, cette fonction est la fonction principale du programme (la première appelée lors de son lancement). Elle crée la fenêtre "window" et son moteur de rendu "renderer" pour les affichages, charge l'image "butterfly" et crée sa texture pour le rendu. Puis, elle gère l'affichage des animations ainsi que les évènements dans une boucle infinie "while (1)".

```
unsigned int y_ligne; // V2
```

Sous "int main(...)", cette ligne déclare la variable locale "y_ligne" de type entier non signé (unsigned int), donc positive. Elle servira de position verticale (y) pour le dessin des lignes horizontales colorées.

```
int couleur_ligne, couleur_coeff, couleur_debut=0, coeff_debut=1; // V2
```

Cette ligne déclare des variables locales de type entier signé (int), donc positives ou négatives.

- **couleur_ligne** : variable stockant la couleur d'une ligne dessinée.
- **couleur_coeff** : variable stockant le coefficient (1 ou -1) pour la couleur des lignes dessinées, qui iront progressivement du noir à l'orange (1), puis de l'orange au noir (-1), etc.
- **couleur_debut=0** : variable stockant la couleur de la première ligne dessinée (initialisée à 0 pour noir).
- **coeff_debut=1** : variable stockant le coefficient (1 ou -1) pour la couleur de la première ligne dessinée (initialisée à 1).

```
SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO); // V2
```

Cette fonction (déjà présente) initialise la bibliothèque SDL2 pour utiliser la vidéo (SDL_INIT_VIDEO) et maintenant aussi l'audio (SDL_INIT_AUDIO).

```
window = SDL_CreateWindow("Butterfly 2", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, MAXW, MAXH, SDL_WINDOW_SHOWN); // V2
```

Cette fonction (déjà présente) crée notre fenêtre "window". Son titre "Butterfly" est modifié en "Butterfly 2" (pour cette deuxième version).

```
SDL_SetColorKey(butterfly, 1, SDL_MapRGB(butterfly->format, 0, 0, 0)); // V2
```

Juste après le chargement de l'image "Butterfly.bmp" en objet surface "butterfly" (fonction SDL_LoadBMP), cette fonction SDL_SetColorKey définit la couleur transparente de l'image, le noir qui entoure notre papillon bleu. Ainsi, les barres horizontales colorées défileront sous le papillon sans être chevauchées par ce noir (devenu transparent).

- **butterfly** : la surface sur laquelle on applique la transparence.
- **1** : active la transparence.
- **SDL_MapRGB(butterfly->format, 0, 0, 0)** : convertit la couleur noire RGB "0, 0, 0" en une valeur entière adaptée au format de "butterfly". C'est cette valeur qui est ensuite utilisée comme couleur transparente pour la surface "butterfly".

```
Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 1024); // V2
```

Après la définition des éléments graphiques, cette fonction initialise le système audio de SDL_mixer avec les paramètres spécifiés.

- **44100** : fréquence d'échantillonnage (en hertz) qui définit la qualité audio.
- **MIX_DEFAULT_FORMAT** : format audio par défaut (16 bits).
- **2** : nombre de canaux (2 = stéréo).
- **1024** : taille du "buffer" audio (sa mémoire tampon).

Ces paramètres sont un bon compromis pour la plupart des systèmes.

```
musique = Mix_LoadMUS("SpaceLam3rs.mp3"); // V2
```

Cette fonction charge le fichier de notre musique "SpaceLam3rs.mp3" en objet "musique" (déclaré en début de programme). Gère plusieurs formats : MP3, WAV, OGG, MIDI,... et même les MOD & MED Amiga ! :)

```
Mix_PlayMusic(musique, -1); // V2
```

À partir de là, cette fonction lance la musique, jouée en boucle par SDL_mixer grâce au paramètre -1.

Reste à gérer dans la boucle infinie "while (1)" l'affichage des barres horizontales colorées ("raster bars") qui doivent défiler sous le papillon.

Pour rappel, cette boucle gère le renderer et donc les animations dans la fenêtre. Elle n'est quittée que par la frappe de la touche "Échap" ou par la fermeture de la fenêtre. Son code est entre accolades {}.

Le principe de ce genre de traitement est toujours le même avec SDL2, pour chaque cycle :

- Gérer les évènements éventuellement survenus (clavier, souris,...).
- Effacer le renderer.
- Déterminer les nouveaux éléments graphiques pour le renderer (position et transformation des textures, dessin d'autres éléments).
- Mettre à jour l'affichage de la fenêtre avec tout ce qui a été rendu sur le renderer.

À grande vitesse, cela produit les animations (avec la prise en compte rapide des éventuels évènements).

Pour faire défiler les barres horizontales colorées sous le papillon

À chaque cycle, on va dessiner des lignes horizontales d'un pixel et de la largeur de la fenêtre, de la première ligne à la dernière ligne de la fenêtre.

À partir d'une couleur de début pour la première ligne, chaque ligne aura une couleur différente de la précédente, de sorte qu'on passe - sur plusieurs lignes - du noir à l'orange, puis de l'orange au noir, etc. Remplissant ainsi la fenêtre de barres horizontales, chacune dégradée verticalement en noir/orange/noir.

Pour que les lignes défilent verticalement dans la fenêtre, il suffit de modifier - à chaque cycle - la couleur de la première ligne, de la même façon : progressivement du noir à l'orange, puis de l'orange au noir, etc.

Dans le cycle, ces barres horizontales colorées doivent être affichées avant l'affichage du papillon, pour être sous celui-ci.

Ce qui donne en termes de code

```
couleur_ligne = couleur_debut; // V2
```

En début de cycle (après SDL_RenderClear), initialise la couleur d'une ligne "couleur_ligne" avec la couleur de début "couleur_debut" (elle-même initialisée à 0/noir au lancement du programme). C'est la couleur de la première ligne.

```
couleur_coeff = coeff_debut; // V2
```

Initialise le coefficient (1 ou -1) pour la couleur des lignes "couleur_coeff" avec le coefficient de début "coeff_debut" (lui-même initialisé à 1 au lancement du programme). C'est le coefficient de la première ligne. Pour rappel, ces coefficients permettent de passer progressivement du noir à l'orange (1), puis de l'orange au noir (-1), etc.

```
for (y_ligne=0; y_ligne<MAXH; y_ligne++) { // V2
```

Cette boucle "for" permet de faire varier la variable "y_ligne" de 0 (y_ligne=0) à MAXH-1/600-1 (y_ligne<MAXH), de 1 en 1 pixel (y_ligne++). Pour rappel, cette variable sert de position verticale (y) pour le dessin des lignes horizontales colorées, donc de la première ligne à la dernière ligne de la fenêtre. Le code de cette boucle "for" est entre accolades {}.

```
SDL_SetRenderDrawColor(renderer, couleur_ligne, couleur_ligne*0.55, 0, 0); // V2
```

Cette fonction définit la couleur utilisée pour dessiner dans le renderer, c'est-à-dire en RGB "couleur_ligne, couleur_ligne*0.55, 0" (le dernier 0 est inexploité ici). En faisant varier "couleur_ligne" de 0 à 255, on obtient donc toutes les couleurs du noir à l'orange (en dégradé), et inversement de l'orange au noir ("couleur_ligne" de 255 à 0).

```
SDL_RenderDrawLine(renderer, 0, y_ligne, MAXW, y_ligne); // V2
```

Cette fonction dessine la ligne horizontale dans le renderer avec la couleur précédemment définie.

- **0, y_ligne** : coordonnées (x, y) de début.
- **MAXW (800), y_ligne** : coordonnées (x, y) de fin.

Précision : coordonnées déterminées par rapport au coin supérieur gauche de la fenêtre.

```
couleur_ligne = couleur_ligne + couleur_coeff * 5; // V2
```

Pour passer à la couleur suivante, incrémente la couleur de ligne "couleur_ligne" de la valeur "couleur_coeff * 5". Si "couleur_coeff" vaut 1, progresse du noir à l'orange (pour -1 progresse de l'orange au noir). 5 étant le facteur de progression dans le dégradé de couleur, plus ce facteur est élevé moins les barres horizontales colorées seront étirées dans la hauteur (et inversement).

```
if (couleur_ligne >= 255 || couleur_ligne <= 0) couleur_coeff = -couleur_coeff; // V2
```

Si la couleur de ligne "couleur_ligne" est supérieure ou égale à 255, ou inférieure ou égale à 0, alors le coefficient pour la couleur des lignes "couleur_coeff" est inversé (pour 1 progresse du noir à l'orange, pour -1 progresse de l'orange au noir).

Après cette boucle "for"

```
couleur_debut = couleur_debut + coeff_debut * 15; // V2
```

Avant le cycle suivant et pour le défilement des barres horizontales colorées, incrémente la couleur de début "couleur_debut" (pour la première ligne) de la valeur "coeff_debut * 15". Si "coeff_debut" vaut 1, progresse du noir à l'orange (pour -1 progresse de l'orange au noir). 15 étant le facteur de progression dans le dégradé de couleur, plus ce facteur est élevé plus les barres horizontales colorées défileront vite verticalement (et inversement).

```
if (couleur_debut >= 255 || couleur_debut <= 0) coeff_debut = -coeff_debut; // V2
```

Si la couleur de début "couleur_debut" est supérieure ou égale à 255, ou inférieure ou égale à 0, alors le coefficient de début "coeff_debut" (pour la première ligne) est inversé (pour 1 progresse du noir à l'orange, pour -1 progresse de l'orange au noir).

Fin des ajouts dans le code source.

Compilation et exécution du programme

Une fois le fichier du code source enregistré (avec le nom "Butterfly2.c"), il faut le compiler pour obtenir son fichier exécutable. Il faudra également le recompiler après chaque modification du code source.

Pour cela, ouvrir un terminal Shell et aller dans le répertoire contenant ce fichier (à l'aide de la commande de changement de répertoire "cd").

Pour rappel, les fichiers de l'image "Butterfly.bmp" et de la musique "SpaceLam3rs.mp3" doivent être présents dans ce même répertoire (vérifier avec la commande "dir").

Puis taper la commande suivante pour le compiler :

```
gcc Butterfly2.c `sdl2-config --cflags --libs` -lSDL2_mixer -o Butterfly2
```

- **gcc** : commande du compilateur GCC permettant de compiler un fichier de code source C en fichier exécutable.
- **Butterfly2.c** : nom du fichier source à compiler.
- **`sdl2-config --cflags --libs`** : paramètre de compilation nécessaire pour utiliser la bibliothèque SDL2 (attention au symbole accent grave " ` ", ce n'est pas une apostrophe " ' ").
- **-lSDL2_mixer** : option indiquant au compilateur de lier le programme à la bibliothèque audio SDL2_mixer.
- **-o** : option permettant de spécifier le nom du fichier exécutable généré.
- **Butterfly2** : nom du fichier exécutable généré (sans extension).

Il n'y a pas de message d'erreur quand la compilation se passe bien. Dans le cas contraire, corriger le code source et recompiler.

Pour lancer le programme, il suffit de taper le nom de son fichier exécutable "Butterfly2", ou de double-cliquer sur son icône dans le dossier concerné de l'interface graphique.

Et vous devriez voir apparaître la fenêtre avec notre joli morpho bleu qui papillonne à l'écran, par-dessus les barres horizontales colorées qui défilent, le tout agrémenté de musique. :)

Précision :

L'animation doit être fluide. Si elle est saccadée, relancer le programme après avoir décoché l'option "Affichage amélioré" ("Enhanced display" en anglais) en éditant votre écran dans les préférences de MorphOS.

Code source complet

Pour conclure le tout, voici le code source complet du programme (les modifications apportées ont le commentaire "// V2" et leur ligne est en gras) :

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_mixer.h> // V2

#define MAXW 800
#define MAXH 600

SDL_Window *window;
SDL_Renderer *renderer;
SDL_Surface *butterfly;
SDL_Texture *texture_butterfly;
Mix_Music *musique; // V2

void fin() {
   if (musique) Mix_FreeMusic(musique); // V2/
   Mix_CloseAudio(); // V2
   if (texture_butterfly) SDL_DestroyTexture(texture_butterfly);
   if (butterfly) SDL_FreeSurface(butterfly);
   if (renderer) SDL_DestroyRenderer(renderer);
   if (window) SDL_DestroyWindow(window);

   SDL_Quit();

   exit(0);
}

void gestion_evenements() {
   SDL_Event event;

   while (SDL_PollEvent(&event)) {
      switch (event.type) {
         case SDL_KEYDOWN:
            if (event.key.keysym.sym == SDLK_ESCAPE) fin();
            break;
         case SDL_WINDOWEVENT:
            if (event.window.event == SDL_WINDOWEVENT_CLOSE) fin();
            break;
      }
   }
}

int main(int argc, char *argv[]) {
   unsigned int y_ligne; // V2
   int couleur_ligne, couleur_coeff, couleur_debut=0, coeff_debut=1; // V2
   unsigned int butterfly_w=200, butterfly_h=113, vitesse=10;
   int x=0, y=0, sens_x=vitesse, sens_y=vitesse, butterfly_l=butterfly_w, battements=20;

   SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO); // V2

   window = SDL_CreateWindow("Butterfly 2", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, MAXW, MAXH, SDL_WINDOW_SHOWN); // V2
   renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
   SDL_SetRenderDrawColor(renderer, 0, 0, 0, SDL_ALPHA_OPAQUE);

   butterfly = SDL_LoadBMP("Butterfly.bmp");
   SDL_SetColorKey(butterfly, 1, SDL_MapRGB(butterfly->format, 0, 0, 0)); // V2
   texture_butterfly = SDL_CreateTextureFromSurface(renderer, butterfly);
   SDL_Rect srcrect_butterfly = { 0, 0, butterfly_w, butterfly_h };

   Mix_OpenAudio(44100, MIX_DEFAULT_FORMAT, 2, 1024); // V2
   musique = Mix_LoadMUS("SpaceLam3rs.mp3"); // V2
```

```
   Mix_PlayMusic(musique, -1); // V2

   while (1) {
      gestion_evenements();

      SDL_RenderClear(renderer);

      couleur_ligne = couleur_debut; // V2
      couleur_coeff = coeff_debut; // V2

      for (y_ligne=0; y_ligne<MAXH; y_ligne++) { // V2
         SDL_SetRenderDrawColor(renderer, couleur_ligne, couleur_ligne*0.55, 0, 0); // V2
         SDL_RenderDrawLine(renderer, 0, y_ligne, MAXW, y_ligne); // V2
         couleur_ligne = couleur_ligne + couleur_coeff * 5; // V2
         if (couleur_ligne >= 255 || couleur_ligne <= 0) couleur_coeff = -couleur_coeff; // V2
      } // V2

      couleur_debut = couleur_debut + coeff_debut * 15; // V2
      if (couleur_debut >= 255 || couleur_debut <= 0) coeff_debut = -coeff_debut; // V2

      x = x + sens_x;
      y = y + sens_y;
      if (x < 0 || x > MAXW) sens_x = -sens_x;
      if (y < 0 || y > MAXH-butterfly_h) sens_y = -sens_y;

      butterfly_l = butterfly_l + battements;
      if (butterfly_l < 0 || butterfly_l > butterfly_w) battements = -battements;

      SDL_Rect dstrect_butterfly = { x-butterfly_l/2, y, butterfly_l, butterfly_h };
      SDL_RenderCopy(renderer, texture_butterfly, &srcrect_butterfly, &dstrect_butterfly);

      SDL_RenderPresent(renderer);
   }
}
```

[Retour en haut] / [Retour aux articles]