Chapitre VI : Sprites

par Loka

Date de publication : 08/05/2006

Dernière mise à jour : 08/05/2006

VI - Sprites

VI-A - Introduction

Qu'est ce qu'un sprite ?

Un sprite désigne une image qui peut être déplacée par rapport au fond de l'écran.

L'usage des sprites est une technique fondamentale dans bien des jeux vidéo en 2 dimensions (par exemple, Mario, Arkanoid, Pac-Man, etc.).

En général, un sprite est une image rectangulaire qui comprend une couleur transparente.

Lorsque le sprite est affiché à l'écran, tout ce qui est derrière un point du sprite de couleur transparente reste visible et les autres sont masqués. Ainsi, le sprite peut avoir une silhouette bien détaillée qui se découpe bien sur le fond de l'écran.

Un exemple de sprite particulièrement courant : le pointeur de la souris de votre ordinateur personnel.

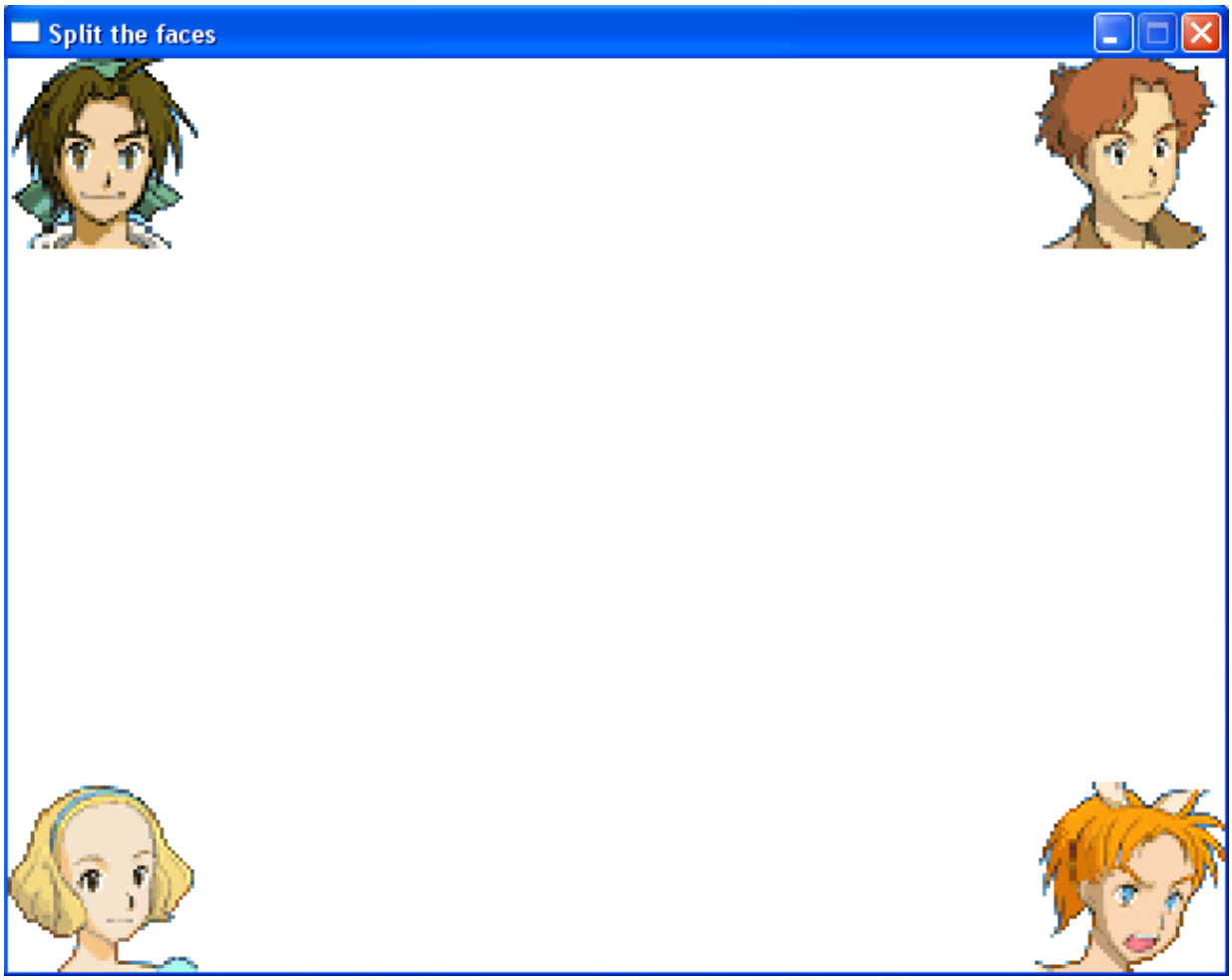
VI-B - Feuille de sprite

Dans cette partie, nous allons voir comment, à partir d'une feuille de sprites, couper un sprite afin de pouvoir l'utiliser individuellement.

Nous allons utiliser une *feuille* composée de quatre sprites :

Afin d'illustrer ce chapitre, nous allons tenter de découper chaque sprite et de les disposer dans chacun des quatre coins de notre fenêtre SDL.

Le résultat final devra ressembler à ceci :



Comme à chaque fois, nous allons commencer par déclarer nos variables :

variables

```
//Les attributs de l'écran (640 * 480)
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;
const int SCREEN_BPP = 32;

//Les attributs de la feuille de sprites (200 * 200)
const int SHEET_WIDTH = 200;
const int SHEET_HEIGHT = 200;

//Les surfaces
SDL_Surface *faces = NULL;
SDL_Surface *screen = NULL;

//La structure d'événements
SDL_Event event;

//Les différentes parties de la feuille de sprites qui vont être blittés
SDL_Rect clip[ 4 ];
```

Nous utilisons, comme d'habitude, des variables globales pour les attributs de l'écran.

De nouvelles variables globales font leur apparition, ce sont les dimensions de notre feuille de sprites.

Pour les surfaces, il y a comme toujours la surface *screen*, mais nous avons en plus une surface *faces* qui va nous permettre de charger et de "manipuler" notre feuille de sprites.

Comme toujours, nous retrouvons notre structure d'événements *event*.

En dernier, nous avons un tableau de quatre **SDL_Rect**. Il va garder les positions et les dimensions de chacun des quatre sprites.

Nous allons maintenant légèrement modifier notre fonction **apply_surface()** afin de pouvoir gérer les sprites qu'on a découpé.

Voyons voir ça de plus près :

apply_surface
```
void apply_surface( int x, int y, SDL_Surface* source, SDL_Surface* destination, SDL_Rect* clip =
NULL )
{
    SDL_Rect offset;

    offset.x = x;
    offset.y = y;

    //On blitte la surface
    SDL_BlitSurface( source, clip, destination, &offset );
}
```

Voici donc notre nouvelle fonction apply_surface().

Le nouvel argument est le SDL_Rect nommé *clip* qui définit la partie rectangulaire de la surface que nous souhaitons blitter.

Nous mettons par défaut cet argument à NULL afin de permettre l'utilisation de cette fonction comme avant (c'est à dire qu'on a la possibilité de ne pas mettre de cinquième argument).

Nous changeons aussi la façon dont est appelé **SDL_BlitSurface()**.

Nous ne mettons plus le second argument à NULL mais nous y mettons l'argument *clip*.

Maintenant SDL_BlitSurface() va *blitter* la région de la surface source définie par *clip*.

Si l'argument est à NULL, SDL_BlitSurface() va *blitter* la surface entière de la source.

Passons au découpage de notre feuille de sprites.

Nous allons définir nos quatre régions rectangulaires ainsi :

clipping
```
	//On coupe la partie en haut à gauche (premier sprite)
	clip[ 0 ].x = 0;
	clip[ 0 ].y = 0;
	clip[ 0 ].w = SHEET_WIDTH/2;
	clip[ 0 ].h = SHEET_HEIGHT/2;

	//On coupe la partie en haut à droite (second sprite)
	clip[ 1 ].x = SHEET_WIDTH/2;
	clip[ 1 ].y = 0;
	clip[ 1 ].w = SHEET_WIDTH/2;
	clip[ 1 ].h = SHEET_HEIGHT/2;

	//On coupe la partie en bas à gauche (troisième sprite)
	clip[ 2 ].x = 0;
	clip[ 2 ].y = SHEET_HEIGHT/2;
	clip[ 2 ].w = SHEET_WIDTH/2;
	clip[ 2 ].h = SHEET_HEIGHT/2;

	//On coupe la partie en bas à droite (quatrième sprite)
	clip[ 3 ].x = SHEET_WIDTH/2;
	clip[ 3 ].y = SHEET_HEIGHT/2;
	clip[ 3 ].w = SHEET_WIDTH/2;
	clip[ 3 ].h = SHEET_HEIGHT/2;
```

Cette partie se fait après avoir tout initialisé.

Nous divisons par 2 car nous avons une feuille de sprites carré possédant quatre sprites (chaque sprite étant de même dimension), deux par ligne et donc il y a aussi deux lignes.

Donc au final, notre découpage définit les régions ainsi :

Nous séparons donc les sprites individuellement depuis la feuille de sprites.

Commençons par rendre l'écran blanc :

Fill screen
```
	//On remplit l'écran de blanc
	SDL_FillRect( screen, &screen->clip_rect, SDL_MapRGB( screen->format, 0xFF, 0xFF, 0xFF ) );
```

Ce que fait SDL_FillRect est de prendre la surface dans le premier argument, et de remplir la région dans le second argument avec la couleur du troisième argument.

La région dans le second argument est la surface *clip_rect* ou la région entière de la surface.

Plaçons maintenant nos sprites dans les quatre coins de notre écran :

application des sprites
```
      //On applique les sprites sur l'écran
      apply_surface( 0, 0, faces, screen, &clip[ 0 ] );
      apply_surface( SCREEN_WIDTH-(SHEET_WIDTH/2), 0, faces, screen, &clip[ 1 ] );
      apply_surface( 0, SCREEN_HEIGHT-(SHEET_HEIGHT/2), faces, screen, &clip[ 2 ] );
      apply_surface( SCREEN_WIDTH-(SHEET_WIDTH/2), SCREEN_HEIGHT-(SHEET_HEIGHT/2), faces, screen,
  &clip[ 3 ] );
```

Comme vous pouvez le constater, nous *blittons* toujours la même surface, la seule différence étant que nous *blittons* une partie différente de la surface à chaque fois.

Je pense qu'il n'y a pas besoin d'expliquer le reste, si vous avez du mal avec les positions par rapport aux variables globales, changer les valeurs et faîtes les calculs, vous verrez qu'il n'y a rien de sorcier.

Maintenant, quand vous aurez beaucoup d'images que vous souhaitez utiliser, il vous suffira de créer des feuilles de sprites et de *blitter* la portion que vous souhaitez à chaque fois.

Cela permet aussi de regrouper les images d'un même *groupe*, par exemple un même sprite sous différents angles de vue, etc... Voici un exemple de feuille de sprite, avec un personnage unique (en deux "couleurs"):

Cette feuille de sprite permet de simuler le déplacement de ce personnage dans toutes les directions (en ne considérant que quatre directions).

Nous reviendrons sur le déplacement d'un perso dans la partie mouvement (Chapitre X) de ce tutoriel.

Version pdf (117 ko - 8 pages)

Télécharger les sources (157 ko)