Chapitre IV : Gestion des événements avec SDL

par Loka

Date de publication : 30/03/2006

Dernière mise à jour : 17/04/2006

La gestion des événement est un passage obligatoire pour qui veut faire des applications SDL.

IV - Gestion des événements
IV-A - Principe
IV-B - Evénement clavier
IV-C - Statut d'une touche

IV - Gestion des événements

Un événement est simplement quelque chose qui se passe. Cela peut être la touche qu'on presse, le mouvement de la souris, le redimensionnement de la fenêtre ou la fermeture de la fenêtre par le X.

IV-A - Principe

Cette partie vous apprendra comment vérifier et attraper un événement, spécialement quand l'utilisateur souhaites quitter sa fenêtre avec le X.

Nous allons donc construire une application qui ferme notre fenêtre quand l'utilisateur clique sur le X de notre fenêtre.

Commençons donc :

header, variables et constantes
```
//Les fichiers d'entête dont on a besoin
#include "SDL/SDL.h"
#include "SDL/SDL_image.h"
#include <string>

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

//Les surfaces
SDL_Surface *image = NULL;
SDL_Surface *screen = NULL;
```

Je ne pense pas avoir besoin de vous expliquer cette partie, elle est la même que dans les chapitres d'avant.

Par contre une nouvelle variable va venir se faufiler dans tout ça :

SDL_Event
```
//La structure d'événements qu'on va utiliser
SDL_Event event;
```

Une structure SDL_Event stocke toutes les données d'événement pour nous afin que nous puissions les utiliser.

Nous ne présenterons pas les fonctions de chargement d'images, d'initialisation de SDL, de chargement de surfaces et de "nettoyage" car ce sont exactement les mêmes que dans les chapitres précédents.

Nous allons donc nous attarder plus sur le main, qui va traiter les événements.

debut main
```
int main( int argc, char* args[] ) {
        //Un moyen de s'assurer que le programme va nous attendre pour quitter
        bool quit = false;
```

Vous vous demandez sûrement ce que fait ici ce booléen, la réponse se trouve à la suite de ce tutorial.

La suite du main est la même que les fois d'avant avec l'appel d'init(), de load_file(), de apply_surface() et de la fonction SDL_Flip().

Ce qui nous intéresse se trouve juste après :

```
        //Tant que l'utilisateur ne veut pas quitter
        while( quit == false ) {
```

Notre booléen sert à ce niveau là.

Cette boucle, qui sera notre boucle principale, tournera jusqu'à ce que l'utilisateur souhaite quitter notre fenêtre.

Il faut donc, dans cette boucle, lorsque l'utilisateur click sur le X de notre fenêtre, qu'on passe notre booléen à **true**.

Ainsi nous quitterons notre boucle principale et nous exécuterons la suite de notre programme qui quittera l'application.

Voyons voir ça de plus près :

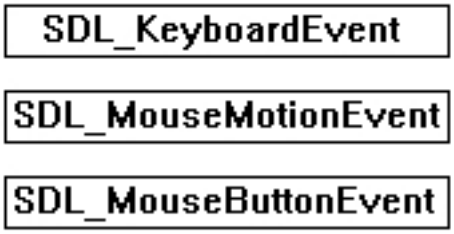
SDL_PollEvent

```
        //Tant qu'il y a un événement à traiter
        while( SDL_PollEvent( &event ) ) {
```

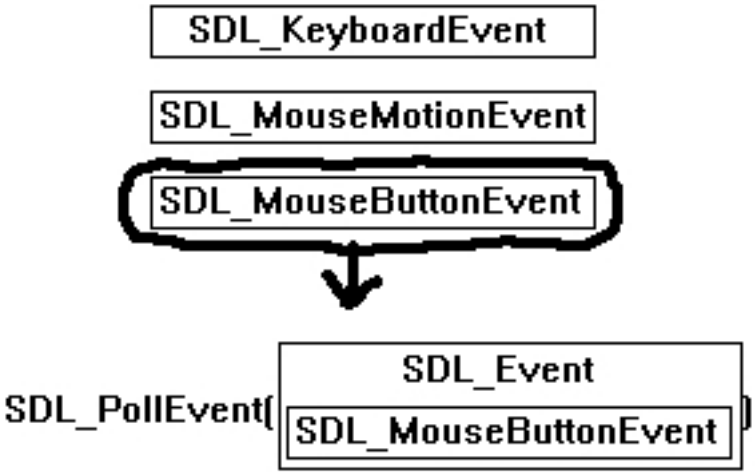
Dans SDL, quand un événement arrive, celui-ci est mis dans une file d'événements de type FIFO (First In, First Out).

Cette file d'événements contient les données de chaque événement qui s'est passé.

Donc si vous pressez sur le bouton de la souris, bougez la souris, puis appuyez sur une touche de votre clavier, la file d'événements devrait ressembler à ceci :



Ce que fait SDL_PollEvent() c'est de prendre un événement de la file et de coller ces données dans notre structure d'événement :



Cette partie du code permet donc de mettre les données des événements dans notre structure tant qu'il y a des événements dans notre file d'événements.

Cela nous permet donc de pouvoir les manipuler par la suite.

Il nous faut maintenant vérifier la nature de l'événement.

Ce que nous cherchons ici est l'événement de fermeture de fenêtre à partir du X, cet événement a pour nom **SDL_QUIT**.

SDL_Quit
```
                        //Si l'utilisateur a cliqué sur le X de la fenêtre
                        if( event.type == SDL_QUIT ) {
                                //On quitte le programme
                                quit = true;
                        }
                }
        }
```

Quand l'utilisateur va cliquer sur le X de notre fenêtre, le type de l'événement sera donc SDL_QUIT comme nous l'avons vu plus haut.

Seulement cela ne fermera pas notre fenêtre pour autant, tout ce que ça fait c'est de nous informer que l'utilisateur souhaite quitter le programme.

Nous voulons donc quitter le programme quand l'utilisateur click sur le X de notre fenêtre, nous mettons donc notre variable booléenne à true, ce qui arrête la boucle dans laquelle nous somme.

il nous suffit ensuite de nettoyer et de faire un return pour quitter.

Netoyage
```
        //On libère les surfaces et on quitte sdl
        clean_up();

        return 0;
}
```

Voila, si vous souhaitez récupérer le code source en entier c'est ici

IV-B - Evénement clavier

Vous savez maintenant comment marche un événement en SDL et vous en avez fait l'expérience en manipulant l'événement SDL_Quit.

Maintenant, dans cette partie, nous allons apprendre à détecter quand une touche est pressée ainsi que comment vérifier quelle touche a été pressée en particulier.

Afin de nous avertir qu'une touche particulière a été pressée, nous allons utiliser créer des surfaces (messages qui nous indiqueront quelle touche à été pressée).

generation des surfaces de message
```
//Génération des surfaces de message
up = TTF_RenderText_Solid( font, "Haut a été pressé.", textColor );
down = TTF_RenderText_Solid( font, "Bas a été pressé.", textColor );
left = TTF_RenderText_Solid( font, "Gauche a été pressé", textColor );
right = TTF_RenderText_Solid( font, "Droite a été pressé", textColor );
```

Comme vous le voyez, ici nous allons utiliser SDL_ttf, donc n'oubliez pas de l'inclure dans les fichiers d'entête.

Ces 4 messages sont générés après avoir tout initialisé et chargé.

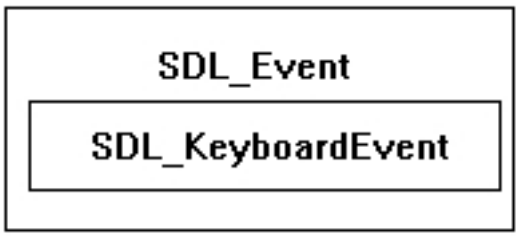
Si vous voulez faire ça correctement, vous devrez vérifier s'il y a une erreur à chaque rendu de texte.

Maintenant nous allons vérifier si une touche est pressée, nous vérifions cela grâce à l'événement SDL_KEYDOWN :

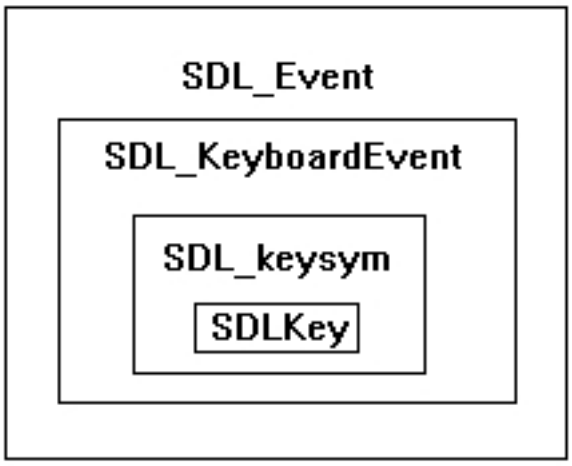
SDL_PollEvent
```
//S'il y a un événement à manipuler
if( SDL_PollEvent( &event ) ) {
        //Si une touche est pressée
        if( event.type == SDL_KEYDOWN ) {
```

Maintenant si une touche est pressée, il nous faut savoir laquelle en particulier afin d'afficher le bon message.

SDL_Poll_event() met les données de **SDL_KEYDOWN** dans la structure d'événement comme étant un SDL_KeyboardEvent nommé *key*



Et dans *key* il y a une structure *keysym*



Et dans la structure *keysym* se trouve enfin le SDL_Key nommé *sym*, qui est la touche qui a été pressée.

Si la touche flèche-haut a été pressée, le *sym* sera **SDLK_UP** et nous afficherons le message up, si c'est la touche fleche-bas qui est pressée, le *sym* sera **SDLK_DOWN** et nous afficherons le message down, etc...

Dans la source, nous vérifions aussi si l'utilisateur ferme la fenêtre avec le X et quittons le programme proprement le cas échéant.

Il vous faudra toujours prévoir une façon de quitter votre application proprement.

Pour revenir à l'affichage de nos messages, nous allons les afficher au centre de notre fenêtre :

affichage message s'il y en a un
```
    //Si un message a besoin d'être affiché
   if( message != NULL ) {
           //Application de l'image sur la l'écran
           apply_surface( 0, 0, background, screen );
           apply_surface( ( SCREEN_WIDTH - message->w ) / 2, ( SCREEN_HEIGHT - message->h ) / 2,
   message, screen );

           //On met à NULL le pointeur vers la surface message
```

affichage message s'il y en a un
```
            message = NULL;
    }
    //Mise à jour de l'écran
    if( SDL_Flip( screen ) == -1 ) {
            return 1;
    }
```

Quand la surface message pointe sur rien , elle sera NULL et rien ne sera *blitté*.

Quand la surface message est non nulle, on applique les surfaces background et le message centré sur l'écran.

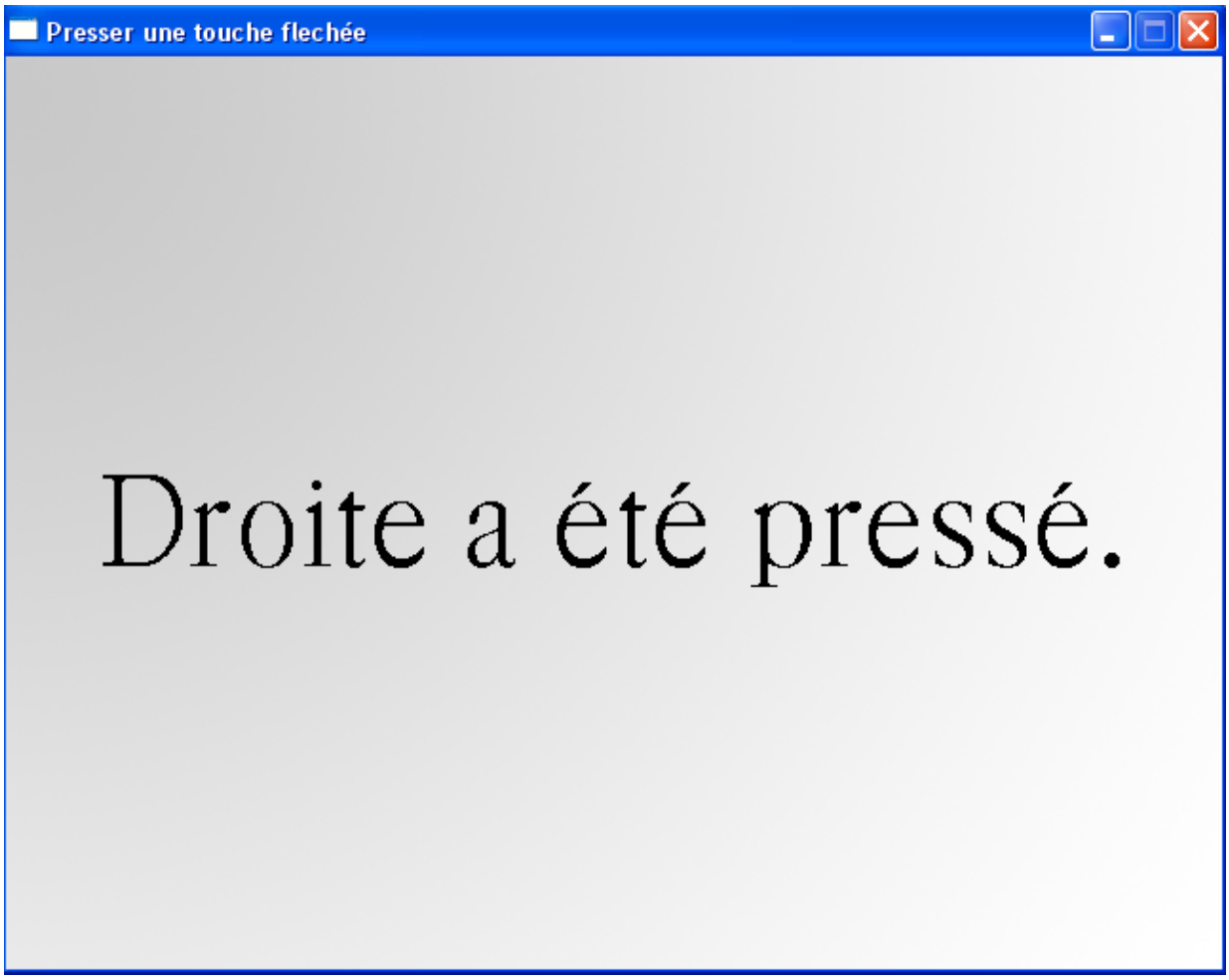
Nous appliquons de nouveau la surface background car quand on met à jour l'écran, il faut tout "reconstruire", c'est à dire qu'il nous faut remettre la surface du fond

La façon de centrer une surface est premièrement de soustraire les hauteur/largeur de la surface qu'on veut blitter à la hauteur/largeur de l'écran sur lequel on veut blitter la surface.

Ainsi, quand la surface est centrée le pas entre les deux côtés est égal, vous divisez la distance restante en deux moitiés égales

Il ne nous reste plus qu'à mettre la surface à NULL et de mettre à jour l'écran.

Si vous appuyez sur droite, vous obtiendrez ceci :



IV-C - Statut d'une touche

Dans cette partie, nous allons vérifier si une touche est pressée sans surveiller dans les événements principaux.

evenements
```
   //Tant que l'utilisateur ne quitte pas
   while( quit == false ) {
           //Tant qu'il y a un événement à manipuler
           while( SDL_PollEvent( &event ) ) {
                   //Si l'utilisateur ferme la fenêtre avec le X
                   if( event.type == SDL_QUIT ) {
                           //On quitte le programme
                           quit = true; }
                   }
```

Comme vous pouvez le constater, Bien que nous cherchons à afficher un message basé sur la touche pressé comme dans la partie précédente, nous ne vérifions aucun événement lié aux touches.

A la place nous allons utiliser **SDL_GetKeyState()** :

SDL_GetKeyState
```
//Récupération du keystates
Uint8 *keystates = SDL_GetKeyState( NULL );
```

Ce que fait **SDL_GetKeyState()** est de nous donner un tableau avec l'état des touches.

Ce tableau est une liste de toutes les touches en précisant si une touche est pressée ou non, comme ceci :

SDLK_UP unpressed
SDLK_DOWN pressed
SDLK_RIGHT unpressed
SDLK_LEFT unpressed
SDLK_INSERT pressed
SDLK_HOME unpressed
SDLK_END pressed
SDLK_PAGEUP unpressed
SDLK_PAGEDOWN unpressed
SDLK_F1 unpressed
SDLK_F2 unpressed
SDLK_F3 unpressed
SDLK_F4 unpressed
SDLK_F5 unpressed

Maintenant nous pouvons donc dire quelle touche est pressée ou non.

Juste pour information, l'argument que nous donnons à **SDL_GetKeyState()** obtient le nombre de clefs disponible.

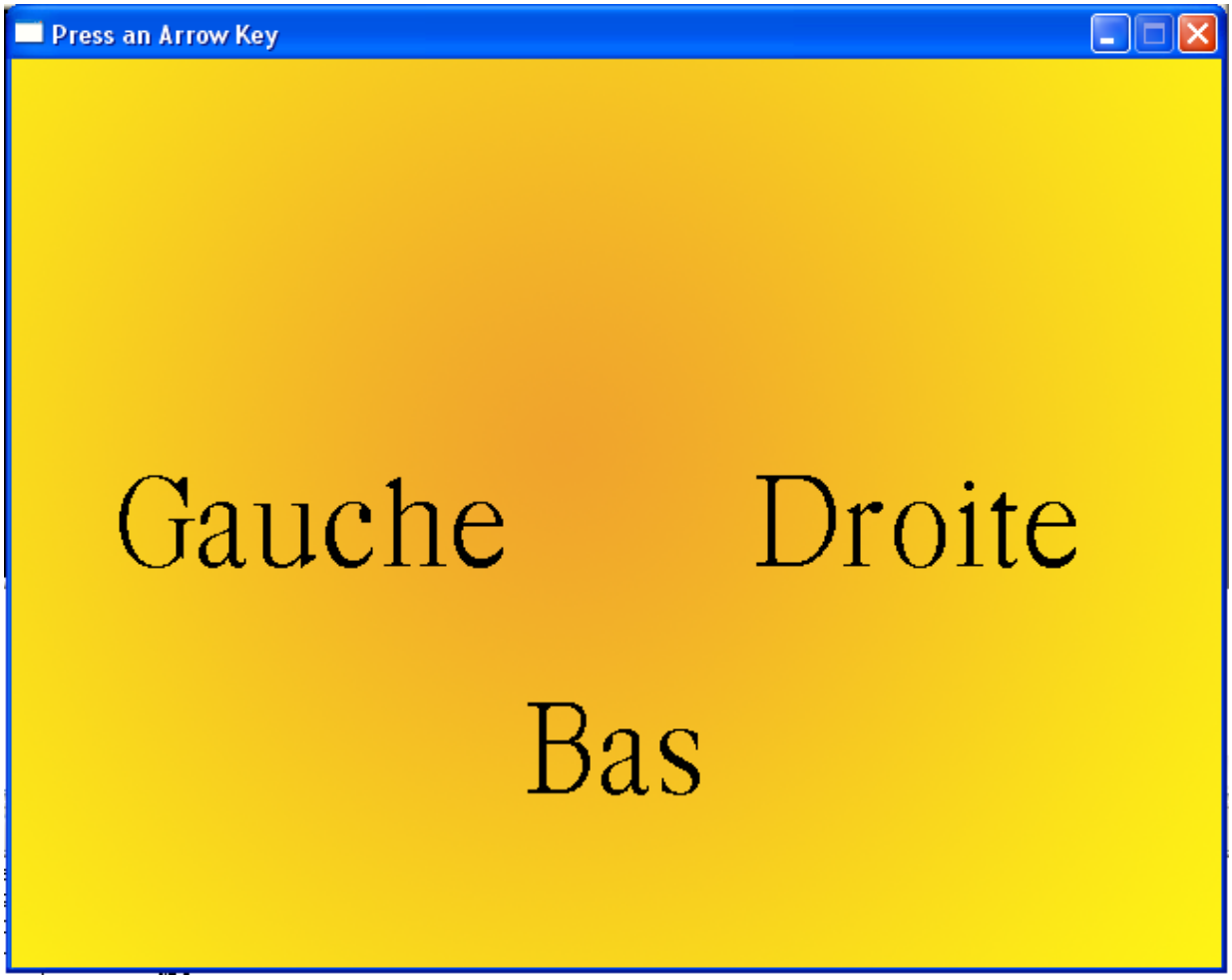
Puisque nous ne nous inquiétons pas de combien de combien de touches il y a, nous le plaçons juste à la NULL.

Maintenant il nous faut afficher les messages, pour cela de simples conditions *if* suffisent :

keystates
```
//Si Haut est pressé
if( keystates[ SDLK_UP ] ) {
        apply_surface( ( SCREEN_WIDTH - up->w ) / 2, ( SCREEN_HEIGHT / 2 - up->h ) / 2, up, screen
);
}
//Si Bas est pressé
if( keystates[ SDLK_DOWN ] ) {
        apply_surface( ( SCREEN_WIDTH - down->w ) / 2, ( SCREEN_HEIGHT / 2 - down->h ) / 2 + (
SCREEN_HEIGHT / 2 ), down, screen );
}
//Si Gauche est pressé
if( keystates[ SDLK_LEFT ] ) {
        apply_surface( ( SCREEN_WIDTH / 2 - left->w ) / 2, ( SCREEN_HEIGHT - left->h ) / 2, left,
screen );
}
//Si Droite est pressé
if( keystates[ SDLK_RIGHT ] ) {
```

keystates
```
        apply_surface( ( SCREEN_WIDTH / 2 - right->w ) / 2 + ( SCREEN_WIDTH / 2 ), ( SCREEN_HEIGHT -
right->h ) / 2, right, screen );
}
//Mise à jour de l'écran
if( SDL_Flip( screen ) == -1 ) {
        return 1;
}
```

Voila le résultat de notre petit application lorsqu'on appuie simultanément sur les flèches bas, gauche et droite :



SDL_GetKeyState() et d'autres fonctions d'état comme SDL_GetModState(), SDL_GetMouseState(), SDL_JoystickGetAxis() et d'autres peuvent être incroyablement utiles.

Apprenez en plus au sujet d'elles dans la référence de l'api SDL, que vous devriez avoir récupéré sur votre ordinateur.

Version pdf

Télécharger les sources des 2 parties

Télécharger les sources du IV-B

Télécharger les sources du IV-C

<= Retour au chapitre III : SDL_TTF

-=Index Tutos SDL=-

=> Accéder au chapitre V : Transparence