SDL2 - Documentation

Fait pour la 2^e année de licence d'informatique, de l'université du Mans.

Fait par
Erwan PECHON.

.

Modifier le 8 février 2023, pour la dernière fois.
Ce document fait 4 page (en plus du sommaire).

Table des matières

I) Bibliothèque nécessaire

Tout fichier contenant de la SDL doit commencer par la ligne : #include <SDL2/SDL.h>.
Cette bibliothèque contient toutes les commandes de bases de la SDL.

Il est aussi possible d'inclure les bibliothèque suivante :

Bibliothèque	⇒utilité
#include <SDL2/SDL_ttf.h>	⇒Permet l'installation de police d'écriture ⇒Permet aussi de simplifier l'affichage de texte.
#include <SDL2/SDL_image.h>	⇒Permet de charger des images de différent type, en tant que surface SDL. (BMP, GIF, JPEG, LBM, PCX, PNG, PNM (PPM/PGM/PBM), QOI, TGA, XCF, XPM, and simple SVG)

II) Initialisé la SDL

Tout programme SDL doit commencé par les lignes suivantes :

"Initialiser"

```
1 if( SDL_Init(Uint32 flags) ){
2       printf( "Erreur SDL_Init : %s\n", SDL_GetError() );
3       return 1;
4 }
```

Cela permet d'activer tout les sous-système dont l'on à besoin. Pour choisir nos sous-système, il faut donner à l'argument flags, l'une des valeurs suivantes (il est possible dans donnée plusieurs, en les séparant par l'opérateur '|') :

Drapeaux	Description
'SDL_INIT_TIMER'	Initialise le système de gestion du temps
'SDL_INIT_AUDIO'	Initialise le système de gestion de l'audio
'SDL_INIT_VIDEO'	Initialise le système de gestion de rendu
'SDL_INIT_JOYSTICK'	Initialise le système de gestion des joysticks
'SDL_INIT_GAMECONTROLLER'	Initialise le système de gestion des contrôleurs de jeux
'SDL_INIT_EVENTS'	Initialise le système de gestion des évènements
'SDL_INIT_EVERYTHING'	Permet de tout initialiser

Après l'initialisation de la SDL, tout erreur doit sauter à la balise 'Quit:', afin de correctement fermer le programme. Pour cela, il faudra utilisé la commande 'goto(Quit);' à la place de la commande 'return(<code erreur>);'.

III) Gérer la fenêtre

1) Paramétrer la fenêtre

Le renderer d'une fenêtre est son contenu. On associe un renderer à une fenêtre, afin de pouvoir calculé l'affichage de la fenêtre, avant de l'afficher, et ainsi éviter des bugs d'affichage.

"Créer une fenêtre et son renderer"

```
1 SDL_Window * window;
2 SDL_Renderer * renderer;
3 if( SDL_CreateWindowAndRenderer(int epaisseurW,int hauteurW,Uint32 flagsW,&window,&
      renderer) ){
4       printf( "Erreur SDL_CreateWindowAndRenderer : %s\n", SDL_GetError() );
5       goto(Quit);
6 }
7 SDL_SetWindowTitle( window , "Titre de la fenêtre" );
```

L'argument flagsW, sert à paramétrer la fenêtre. Il faut lui donner l'une des valeurs suivantes (il est possible dans donnée plusieurs, en les séparant par l'opérateur '|') :

Drapeaux	Description
'SDL_WINDOW_FULLSCREEN'	Crée une fenêtre en plein écran
'SDL_WINDOW_FULLSCREEN_DESKTOP'	Crée une fenêtre en plein écran à la résolution du bureau
'SDL_WINDOW_SHOWN'	Crée une fenêtre visible
'SDL_WINDOW_HIDDEN'	Crée une fenêtre non visible
'SDL_WINDOW_BORDERLESS'	Crée une fenêtre sans bordures
'SDL_WINDOW_RESIZABLE'	Crée une fenêtre redimensionnable
'SDL_WINDOW_MINIMIZED'	Crée une fenêtre minimisée
'SDL_WINDOW_MAXIMIZED'	Crée une fenêtre maximisée

Il faut absolument les détruire à la fin du programme avec :

"Détruire une fenêtre et son renderer"

```
Quit:
	if( renderer ) SDL_DestroyRenderer( renderer );
	if( window ) SDL_DestroyWindow( window );
	SDL_Quit();
	return 0;
```

2) Gestion des événement

A) Les événements possibles

Type d'évènements	Valeur du champ '<type>'	Champ de 'SDL_Event' correspondant	Description
Événements de l'application	'SDL_QUIT'	'quit'	Demande de fermeture du programme
Évènements de la fenêtre	'SDL_WINDOWEVENT'	'window'	Changement d'état de la fenêtre
	'SDL_SYSWMEVENT'	'syswm'	Évènement dépendant du sytème
Événements du clavier	'SDL_KEYDOWN'	'key'	Une touche est pressé
	'SDL_KEYUP'	'key'	Une touche est relâchée
	'SDL_TEXTEDITING'	'edit'	Édition de texte
	'SDL_TEXTINPUT'	'text'	Saisie de texte
Événements de la souris	'SDL_SDL_MOUSEMOTION'	'motion'	Déplacement de la souris
	'SDL_MOUSEBUTTONDOWN'	'button'	Une touche de la souris est pressée
	'SDL_MOUSEBUTTONUP'	'button'	Une touche de la souris est relâchée
	'SDL_MOUSEWHEEL'	'wheel'!'	La molette est utilisée

B) Lire un événement

Pour lire un événement, nous avons 3 fonctions, que l'ont utilise de la façon suivante :

"Lire un événement"

```
SDL_Event event;
if( <fonction>(&event) ){
	switch(event.type){
		case <evenement> :
			<traitement de l'événement>
		case <evenement2> :
			<traitement de l'événement2>
		...
	}
}
```

Attendre qu'un événement arrive La commande 'SDL_WaitEvent(&event);' bloque l'exécution du programme, jusqu'à ce qu'un événement arrive, peut importe lequel.
Cette commande renvoi 0 en cas d'erreur et 1 sinon.

Attendre qu'un événement arrive, pendant un temps limité La commande 'SDL_WaitEventTimeout(&event, int time);' bloque l'exécution du programme, jusqu'à :

- ce qu'un événement arrive, peut importe lequel.
- ce que <time> milliseconde ce soit passé.

Cette commande renvoi 0 en cas d'erreur, ou si aucun événements n'est arrivé avant <time> milliseconde et 1 sinon.

Vérifié si un événement est dans la pile La commande 'SDL_PollEvent(&event);' test si il y à des événements dans la file, puis continue.
Cette commande renvoi 1 si elle a lu un événement, et 0 sinon.
Attention, dans ce cas, il faut traité tout les événement de la file d'un seul coup. Il faut donc utilisé un while au lieu d' un if.

3) Affichage du contenu

Comme dit précédemment, le contenu est préparer dans le renderer, avant de l'afficher à la fenêtre. Il y à cependant des moyens de préparer plusieurs contenu à l'avance.

Les viewports voir 'SDL_RenderSetViewport'

Les display voir 'SDL_SetWindowPosition'

IV) Dessiner

Tout les dessins doivent être fait sur le renderer. Il y à plusieurs fonctions de dessins simple dans la SDL :

"Afficher un dessin"

```
1 SDL_PresentRenderer( renderer );
```

"Changer la couleur du pinceau"

```
1 if( SDL_SetRendererDrawColor( renderer , int r , int g , int b , int a) ){
2       printf( "Erreur, la fonction de dessin à planté : %s\n" , SDL_GetError() );
3       goto(Quit);
4 }
```

Les commandes suivantes utiliseront la dernière couleur de pinceau sélectionner.

"Changer le fond d'écran"

```
1 if( SDL_RendererClear( renderer ) ){
2       printf( "Erreur, la fonction de dessin à planté : %s\n" , SDL_GetError() );
3       goto(Quit);
4 }
```

Dessiner des formes géométriques simple