


Fenêtrage OpenGL avec SDL

par Tony BAYART ([Site perso de Tony BAYART](#))

Date de publication : 18/03/2007

Dernière mise à jour : 26/04/2007

OpenGL est avant tout une bibliothèque graphique. La portabilité fait partie des raisons qui rendent cette bibliothèque si populaire. Or pour utiliser OpenGL, il faut lui fournir un contexte de rendu, qui est fourni par le système. Pour conserver cette portabilité, plutôt que d'écrire le code pour différents systèmes, nous allons nous en remettre à une autre bibliothèque portable : la  **SDL**.

- I - Préparatifs
- II - Initialisation de la SDL
- III - Initialisation OpenGL
- IV - La boucle principale
- V - Fin du programme
- VI - Remerciements
- VII - Téléchargements

I - Préparatifs

Avant de nous lancer dans la création d'un programme quelconque utilisant la SDL, il nous faut les fichiers de la bibliothèque. Rendez-vous sur le site de la SDL : <http://www.libsdl.org/>, sélectionnez dans le menu "Download" la dernière version (la **version 1.2** est disponible au moment où j'écris ces lignes) et finalement, dans la section "Development Libraries" téléchargez l'archive adaptée à votre système. L'archive téléchargée contient 3 répertoires qui nous intéressent :

- Le répertoire "include" contient les fichiers ".h" à inclure dans votre programme. Copiez les dans un sous-dossier "SDL" du répertoire "include" de votre compilateur.
- Le répertoire "lib" contient les bibliothèques à lier avec le programme. Copiez les fichiers dans le répertoire "lib" de votre compilateur.
- Enfin, le répertoire bin contient la bibliothèque SDL compilée. Ce fichier devra être copié au même emplacement que le programme voulant l'utiliser.

Une fois ces préparatifs terminés, nous pouvons commencer.

II - Initialisation de la SDL

Se servir de la SDL pour obtenir une fenêtre d'affichage OpenGL ou un affichage plein écran est relativement simple. Il nous faut tout d'abord les prototypes des fonctions de la bibliothèque en incluant son fichier d'entête :

Fichiers à inclure

```
#include <stdlib.h>
#include <iostream>
#include <SDL/SDL.h>
#include <GL/GL.h>
```

Votre compilateur doit bien entendu connaître l'emplacement sur votre disque où se trouve ce fichier. Comme tout programme qui se respecte, il nous faut une fonction principale :

Le main

```
int main( int argc, char** argv )
{
```

La première chose à faire pour utiliser la SDL, c'est de l'initialiser. Ce qui se fait tout simplement avec l'appel à la fonction **SDL_Init**. L'initialisation de la SDL pouvant échouer, la fonction **SDL_Init** retourne un **int** indiquant la réussite ou l'échec de l'initialisation. Le seul paramètre de cette fonction est construit à partir de flags d'initialisation pour activer les différents sous-systèmes de la SDL qui nous intéressent. En ce qui nous concerne, nous n'avons besoin que de l'affichage et utiliserons donc uniquement le flag **SDL_INIT_VIDEO** de cette manière :

Initialisation de la SDL


```
// initialisation de la SDL
if( SDL_Init(SDL_INIT_VIDEO) < 0 )
{
    std::cerr << "Echec SDL_Init : " << SDL_GetError() << std::endl;
    return (EXIT_FAILURE);
}
```


Maintenant que la SDL est initialisée, il faudra également penser à libérer les ressources qu'elle utilise lorsque le programme s'arrêtera. Pour cela, un simple appel à **SDL_Quit** suffit avant de terminer le programme. Une méthode simple pour que **SDL_Quit** soit appelée lorsque notre programme se termine est d'utiliser la méthode **atexit** pour référencer **SDL_Quit** en procédure de sortie :

Procédure de fermeture

```
// SDL initialisée, on active la procédure de sortie en fin de programme
atexit( SDL_Quit );
```


De cette manière, lorsque le programme se terminera, il effectuera automatiquement un appel à **SDL_Quit**.

 *atexit permet de référencer les fonctions à appeler lorsque le programme se termine. Dans notre cas, atexit appellera SDL_Quit lorsque le programme se terminera. Lorsque vous référencez plusieurs fonctions avec atexit, elles seront toutes appelées lorsque le programme se terminera.*

 *Lorsque plusieurs fonctions sont référencées par atexit, celles-ci sont appelées dans l'ordre inverse de leur référencement. Ainsi, la première fonction appelée sera la dernière fonction référencée.*

III - Initialisation OpenGL

La fonction **SDL_GL_SetAttribute** que nous allons maintenant utiliser est spécifique au mode de rendu OpenGL. Elle sert entre autres à paramétrer la précision du "depth buffer" (aussi appelé zbuffer) ou encore activer le "double buffer", ce à quoi nous allons nous limiter pour notre application de base.

 *Dans la version actuelle de la SDL, le double buffer est actif par défaut. Or, même si peu probable, il se peut que cette information change dans une version future. Je vous conseille donc d'appeler `SDL_GL_SetAttribute` pour mettre en place le double buffer, sachant que cela ne vous coûte rien en terme de performances.*

L'activation du double buffer se fait avec la ligne suivante :

Activation du double buffer

```
// assurons nous que le mode "double buffer" est bien demande
SDL_GL_SetAttribute(SDL_GL_DOUBLEBUFFER, 1);
```

Notez que les appels à la fonction **SDL_GL_SetAttribute** sont à faire avant la création de la fenêtre OpenGL. N'hésitez pas à consulter la documentation sur [les attributs OpenGL de la SDL](#) pour connaître tous les paramètres qui vous sont accessibles. Nous pouvons maintenant créer la fenêtre OpenGL à l'aide de la fonction **SDL_SetVideoMode** qui sert justement à activer l'affichage de la SDL.

Mode fenêtré OpenGL

```
// activation de l'affichage SDL en mode fenetre
SDL_Surface* pScreen = SDL_SetVideoMode(640, 480, 32, SDL_OPENGL);
```

Ici on demande à la SDL de nous créer une fenêtre de dimensions **640x480**, respectivement la largeur et la hauteur de la fenêtre, avec une profondeur de couleurs de `{{32 bits}}`. Le 4ème paramètre est construit sur différents drapeaux (*flags* en anglais), mais pour utiliser OpenGL avec la SDL, un seul est nécessaire : **SDL_OPENGL**. Un second drapeau vous intéressera également puisqu'il s'agit de basculer une fenêtre en mode plein écran. Pour cela, il suffit d'ajouter **SDL_FULLSCREEN** au drapeau déjà présent de la manière suivante :

Mode plein écran OpenGL

```
// activation de l'affichage SDL en mode plein ecran
SDL_Surface* pScreen = SDL_SetVideoMode(640, 480, 32, SDL_OPENGL | SDL_FULLSCREEN);
```

Comme la création de la fenêtre peut échouer, ce pour diverses raisons, il est important de vérifier que ce n'est pas le cas et que la fenêtre a effectivement été créée :

Vérification `SDL_SetVideoMode`

```
// verification de l'activation de l'affichage
if( !pScreen )
{
    std::cerr << "Echec de creation de la fenetre en 640x480 : " << SDL_GetError() <<
std::endl;
    return (EXIT_FAILURE);
}
```

De cette manière, l'échec de la création de la fenêtre provoquera l'arrêt de notre programme.

Une fois la fenêtre créée, nous devons nous assurer que le mode double buffer que nous avons demandé est bien actif. Pour cela nous avons accès à **SDL_GL_GetAttribute** qui va récupérer l'information demandée et stocker son

état dans une variable. Comme cette fonction peut échouer, il faut également tester la valeur de retour, ce qui nous mène à procéder de la manière suivante :

Vérification du double buffer

```
// recuperation de l'etat du parametre "double buffer"
int nValue;
if( SDL_GL_GetAttribute(SDL_GL_DOUBLEBUFFER, &nValue) < 0)
{
    std::cerr << "Echec de recuperation du parametre SDL_GL_DOUBLEBUFFER : " << SDL_GetError()
<< std::endl;
    return (EXIT_FAILURE);
}

// assurons nous que le mode "double buffer" est bien actif
if(nValue != 1)
{
    std::cerr << "Erreur : SDL_GL_DOUBLEBUFFER inactif" << std::endl;
    return (EXIT_FAILURE);
}
```

A ce stade, nous avons créés une fenêtre SDL prête à afficher le résultat de nos appels à OpenGL.

IV - La boucle principale

Nous avons notre fenêtre dans laquelle sera visible notre rendu OpenGL. Pour bien faire, il reste à définir la boucle principale dans laquelle le rendu est appelé. Cette dernière traitera également les événements de la SDL pour quitter lorsque l'utilisateur le demandera. Initialisée à **true**, cette variable indique que notre boucle principale est active.

Boucle principale

```
bool bRunning = true;
// boucle principale du programme
while(bRunning)
{
```

La SDL gère les événements tels que par exemple la pression d'une touche, son relâchement, un déplacement de la souris et d'autres événements encore. Nous avons donc besoin d'une boucle pour gérer ces événements. Pourquoi une boucle ? Tout simplement parce qu'un événement n'arrivant pas toujours seul, ils sont stockés dans une pile d'événements. La fermeture de la fenêtre créée par la SDL est également un événement. Nous allons donc créer la boucle suivante :

Traitement des événements

```
SDL_Event event; // un evenement SDL
// recuperation d'un evenement dans la pile
while(SDL_PollEvent(&event))
{
    // analyse du message
    switch(event.type)
    {
        // message demande a quitter
        case SDL_QUIT:
            bRunning = false;
            break;


        // message signalant l'appui sur une touche
        case SDL_KEYDOWN:
            // la touche ESCAPE provoque la fin du programme
            if (event.key.keysym.sym == SDLK_ESCAPE)
                bRunning = false;
            break;
    } // fin du switch
} // fin de la boucle de gestion des evenements
```

A la suite de cette boucle, nous pouvons mettre la suite du programme. Dans notre cas, on va juste effacer le buffer afin d'éviter un scintillement et demander l'échange *front/back* étant donné que nous avons un affichage en "double buffer". C'est la SDL qui gère notre fenêtre et donc notre affichage, nous allons donc faire appel à la fonction **SDL_GL_SwapBuffers** pour l'échange des buffers. Voici donc la fin de la boucle principale :

Rendu OpenGL

```
// effacement de l'ecran
glClear(GL_COLOR_BUFFER_BIT);

// echange des buffers back et front
SDL_GL_SwapBuffers();
} // fin de la boucle principale du programme
```

 **Avertissement aux familiers de la SDL en utilisation 2D classique (fenêtre non OpenGL).** Nous utilisons ici **SDL_GL_SwapBuffers** en lieu et place de la fonction **SDL_Flip** pour échanger les affichages "front" et "back" en mode "double buffer". En effet, **SDL_Flip** ne fonctionne pas lorsque le mode OpenGL est activé et pourrait provoquer des erreurs durant l'exécution. De même, les opérations de transfert de **SDL_Surface** vers la fenêtre

*d'affichage telle que **SDL_BlitSurface** sont à proscrire. Une dégradation importante des performances ou un plantage de l'application risquerait de survenir.*

V - Fin du programme

Et nous voilà arrivés à la fin de notre programme auquel il manque un petit bout de code afin qu'il se termine correctement :

Fin du programme

```
std::cerr << "Fin normale du programme" << std::endl;

// pour eviter les warnings indiquant que argc et argv ne sont pas utilises
(argc);
(argv);

return (EXIT_SUCCESS);
}
```

J'espère que cet article servira à ceux qui veulent débiter avec la SDL et OpenGL. J'ai essayé de faire simple tout en détaillant bien ce petit programme afin de vous initier aux bases. Bien entendu, nous sommes loin d'avoir tout vu de l'utilité de la SDL, que ce soit avec OpenGL ou non. N'hésitez donc pas à parcourir le site officiel et la documentation de cette puissante bibliothèque. Je vous invite également à consulter la base de connaissance de [developpez.com](http://jeux.developpez.com/tutoriels/) concernant la SDL : <http://jeux.developpez.com/tutoriels/>.

VI - Remerciements

Je remercie **Fearyourself** pour son aide précieuse lors de la rédaction de cet article ainsi que pour ses conseils avisés et ses corrections.

VII - Téléchargements

Télécharger les sources de cet article (zip, 203 Ko) ([Mirroir](#))

Version PDF de l'article (51 Ko) ([Mirroir](#))

