

[Accueil](#) > [Cours](#) > [Apprenez à programmer en C !](#) > Installation de la SDL


Apprenez à programmer en C !

40 heures  Moyenne

Mis à jour le 22/01/2021



Installation de la SDL

 [Connectez-vous](#) ou [inscrivez-vous](#) gratuitement pour bénéficier de toutes les fonctionnalités de ce cours !

À partir de maintenant, fini la théorie : nous allons enfin passer au concret ! Dans cette nouvelle et importante partie, nous allons nous faire plaisir et pratiquer grâce à une bibliothèque que l'on appelle la SDL.

Vous avez déjà découvert la plupart des fonctionnalités du langage C, bien qu'il y ait toujours des petits détails complexes et croustillants à découvrir. Ce livre pourrait donc s'arrêter là en annonçant fièrement : « C'est bon, vous avez appris à programmer en C ! ». Pourtant, quand on débute, on n'a en général pas le sentiment de savoir programmer tant qu'on n'est pas « sorti » de la console.

La SDL est une bibliothèque particulièrement utilisée pour créer des jeux en 2D. Nous allons dans ce premier chapitre en apprendre plus sur cette bibliothèque et découvrir comment l'installer.

On dit que la SDL est une « bibliothèque tierce ». Il faut savoir qu'il existe deux types de bibliothèques.

- **La bibliothèque standard** : c'est la bibliothèque de base qui fonctionne sur tous les OS (d'où le mot « standard ») et qui permet de faire des choses très basiques comme des `printf`. Elle a été automatiquement installée lorsque vous avez téléchargé votre IDE et votre compilateur.

Au long des parties I et II, nous avons uniquement utilisé la bibliothèque standard (`stdlib.h` , `stdio.h` , `string.h` , `time.h` ...). Nous ne l'avons pas étudiée dans son intégralité mais nous en

avons vu un assez gros morceau. Si vous voulez tout savoir sur la bibliothèque standard, faites une recherche sur Google, par exemple, en tapant « C standard library », et vous aurez la liste des prototypes ainsi qu'une brève explication de chacune des fonctions.

- **Les bibliothèques tierces** : ce sont des bibliothèques qui ne sont pas installées par défaut. Vous devez les télécharger sur Internet et les installer sur votre ordinateur.

Contrairement à la bibliothèque standard, qui est relativement simple et qui contient assez peu de fonctions, il existe des milliers de bibliothèques tierces écrites par d'autres programmeurs. Certaines sont bonnes, d'autres moins, certaines sont payantes, d'autres gratuites, etc. l'idéal étant de trouver des bibliothèques de bonne qualité et gratuites à la fois !

Je ne peux pas faire un cours pour toutes les bibliothèques tierces qui existent. Même en y passant toute ma vie 24h / 24, je ne pourrais pas !

J'ai donc fait le choix de vous présenter une et une seule bibliothèque écrite en C et donc utilisable par des programmeurs en langage C tels que vous.

Cette bibliothèque a pour nom *SDL*. Pourquoi ai-je choisi cette bibliothèque plutôt qu'une autre ? Que permet-elle de faire ?

Autant de questions auxquelles je vais commencer par répondre.

Pourquoi avoir choisi la SDL ?



Choisir une bibliothèque : pas facile !

Comme je vous l'ai dit à l'instant, il existe des milliers et des milliers de bibliothèques à télécharger. Certaines d'entre elles sont simples, d'autres plus complexes. Certaines sont tellement grosses que même tout un cours comme celui que vous êtes en train de lire ne suffirait pas !

Faire un choix est donc dur. De plus, c'est la première bibliothèque que vous allez apprendre à utiliser (si on ne compte pas la bibliothèque standard), il vaut donc mieux commencer par une bibliothèque simple.

J'ai rapidement constaté que la majorité de mes lecteurs souhaitait découvrir comment ouvrir des fenêtres, créer des jeux, etc. Enfin, si vous aimez la console on peut continuer longtemps, si vous voulez... Non ? Ah bon, tiens c'est curieux !

Quant à moi, non seulement j'ai bien envie de vous montrer comment on peut faire tout ça, mais en plus je tiens absolument à vous faire pratiquer. En effet, nous avons bien fait quelques TP dans les parties I et II, mais ce n'est pas assez. C'est en forgeant que l'on devient forgeron, et c'est en programmant que euh... Bref, vous m'avez compris !

Je suis donc parti pour vous à la recherche d'une bibliothèque à la fois simple et puissante pour que vous puissiez rapidement réaliser vos rêves les plus fous (presque) sans douleur (tout est relatif, bien sûr !).

La SDL est un bon choix !

Nous allons étudier la bibliothèque SDL (fig. suivante). Pourquoi celle-ci et non pas une autre ?



- C'est **une bibliothèque écrite en C**, elle peut donc être utilisée par des programmeurs en C tels que vous. Notez que comme la plupart des bibliothèques écrites en C, il est possible de les utiliser en C++ ainsi que dans d'autres langages.
- C'est **une bibliothèque libre et gratuite** : cela vous évitera d'avoir à investir pour lire la suite du livre. Contrairement à ce que l'on pourrait penser, trouver des bibliothèques libres et gratuites n'est pas très difficile, il en existe beaucoup aujourd'hui. Une bibliothèque libre est tout simplement une bibliothèque dont vous pouvez obtenir le code source. En ce qui nous concerne, voir le code source de la SDL ne nous intéressera pas. Toutefois, le fait que la bibliothèque soit libre vous garantit plusieurs choses, notamment sa pérennité (si le développeur principal arrête de s'en occuper, d'autres personnes pourront la continuer à sa place) ainsi que sa gratuité le plus souvent. La bibliothèque ne risque donc pas de disparaître du jour au lendemain.
- **Vous pouvez réaliser des programmes commerciaux et propriétaires avec.** Certes, c'est peut-être un peu trop vouloir anticiper, mais autant choisir une bibliothèque gratuite qui vous laisse un maximum de libertés. En effet, il existe deux types de bibliothèques libres :
 - les bibliothèques sous **license GPL** : elles sont gratuites et vous pouvez avoir le code source, mais vous êtes obligés en contrepartie de fournir le code source des programmes que vous réalisez avec ;
 - les bibliothèques sous **license LGPL** : c'est la même chose, sauf que cette fois vous n'êtes pas obligés de fournir le code source de vos programmes. Vous pouvez donc réaliser des programmes propriétaires avec.

Bien qu'il soit possible juridiquement de ne pas diffuser votre code source, je vous invite à le faire quand même. Vous pourrez ainsi obtenir des conseils de programmeurs plus expérimentés que vous. Cela vous permettra de vous améliorer.

Après, c'est vous qui choisirez de réaliser des programmes libres ou propriétaires, c'est surtout une question de mentalité. Je ne rentrerai pas dans le débat ici, pas plus que je ne prendrai position : on peut tirer du bon comme du mauvais dans chacun de ces deux types de programmes.

- C'est **une bibliothèque multi-plates-formes**. Que vous soyez sous Windows, Mac ou Linux, la SDL fonctionnera chez vous. C'est même d'ailleurs ce qui fait que cette bibliothèque est impressionnante aux yeux des programmeurs : elle fonctionne sur un très grand nombre de systèmes d'exploitation. Il y a Windows, Mac et Linux certes, mais elle peut aussi fonctionner sur Atari, Amiga, Symbian, Dreamcast, etc. En clair, vos programmes pourraient très bien fonctionner sur de vieilles machines comme l'Atari ! Il faudrait néanmoins faire quelques petites adaptations et peut-être utiliser un compilateur spécial. Nous n'en parlerons pas ici.
- Enfin, la bibliothèque permet de **faire des choses amusantes**. Je ne dis pas qu'une bibliothèque mathématique capable de résoudre des équations du quatrième degré n'est pas intéressante, mais je tiens à ce que ce cours soit ludique autant que possible afin de vous motiver à programmer.

La SDL n'est pas une bibliothèque spécialement conçue pour créer des jeux vidéo. Je l'admets, la plupart des programmes utilisant la SDL sont des jeux vidéo, mais cela ne veut pas dire que vous êtes forcément obligés d'en créer. A priori, tout est possible avec plus ou moins de travail, j'ai déjà eu l'occasion de voir des éditeurs de texte développés à l'aide de la SDL, bien qu'il y ait plus adapté. Si vous souhaitez développer des interfaces graphiques classiques sous forme de fenêtres (boutons, menus, etc.), je vous invite à vous renseigner plutôt sur GTK+.

Les possibilités offertes par la SDL

La SDL est une bibliothèque bas niveau. Vous vous souvenez de ce que je vous avais dit au tout début du cours à propos des langages haut niveau et bas niveau ? Eh bien ça s'applique aussi aux bibliothèques.

- - **Une bibliothèque bas niveau** : c'est une bibliothèque disposant de fonctions très basiques. Il y a en général peu de fonctions car on peut tout faire avec elles. Comme les fonctions restent basiques, elles sont très rapides. Les programmes réalisés à l'aide d'une telle bibliothèque sont donc en général ce qui se fait de plus rapide.
 - **Une bibliothèque haut niveau** : elle possède beaucoup de fonctions capables d'exécuter de nombreuses actions. Cela la rend plus simple d'utilisation.

Toutefois, une bibliothèque de ce genre est généralement « grosse », donc plus difficile à étudier et à connaître intégralement. En outre, elle est souvent plus lente qu'une bibliothèque bas niveau (bien que parfois, ça ne soit pas vraiment visible).

Bien entendu, il faut nuancer. On ne peut pas dire « une bibliothèque bas niveau est mieux qu'une bibliothèque haut niveau » ou l'inverse. Chacun des deux types présente des qualités et des défauts. La SDL que nous allons étudier fait plutôt partie des bibliothèques bas niveau.

Il faut donc retenir que la SDL propose surtout des fonctions basiques. Vous avez par exemple la possibilité de dessiner pixel par pixel, de dessiner des rectangles ou encore d'afficher des images. C'est tout, et c'est suffisant.

- En faisant bouger une image, vous pouvez faire se déplacer un personnage.
- En affichant plusieurs images d'affilée, vous pouvez créer une animation.
- En combinant plusieurs images côte à côte, vous pouvez créer un véritable jeu.

Pour vous donner une idée de jeu réalisable avec la SDL, sachez que l'excellent « Civilization : Call to power » a été adapté pour Linux à l'aide de la bibliothèque SDL (fig. suivante).



Ce qu'il faut bien comprendre, c'est qu'en fait tout dépend de vous et éventuellement de votre équipe. Vous pouvez faire des jeux encore plus beaux si vous avez un graphiste doué sous la main.

La seule limite de la SDL, c'est la 2D. Elle n'est en effet pas conçue pour la 3D. Voici une liste de jeux que l'on peut parfaitement concevoir en SDL (ce n'est qu'une petite liste, tout est possible a priori tant que ça reste de la 2D) :

- Casse-briques ;
- Bomberman ;
- Tetris ;
- jeux de plate-forme : Super Mario Bros, Sonic, Rayman...
- RPG 2D : Zelda, les premiers Final Fantasy, etc.

Il m'est impossible de faire une liste complète, la seule limite ici étant l'imagination. J'ai d'ailleurs vu un des lecteurs de ce cours réaliser un croisement osé entre un casse-briques et un Tetris.

Redescendons sur Terre et reprenons le fil de ce cours. Nous allons maintenant installer la SDL sur notre ordinateur avant d'aller plus loin.

Téléchargement de la SDL



Le [site de la SDL www.libsdl.org](http://www.libsdl.org) devrait bientôt devenir incontournable pour vous. Là-bas, vous trouverez tout ce dont vous avez besoin, en particulier la bibliothèque elle-même à télécharger ainsi que sa documentation.

Sur le site de la SDL, rendez-vous dans le menu à gauche, section **Download** .

Téléchargez la version de la SDL la plus récente que vous voyez (SDL 1.2 au moment où j'écris ces lignes).

La page de téléchargement est séparée en plusieurs parties.

- - **Source code** : vous pouvez télécharger le code source de la SDL. Comme je vous l'ai dit, le code source ne nous intéresse pas. Je sais que vous êtes curieux et que vous voudriez savoir comment c'est fait à l'intérieur, mais actuellement ça ne vous apportera rien. Pire, ça vous embrouillera et ce n'est pas le but.
 - **Runtime libraries** : ce sont les fichiers que vous aurez besoin de distribuer en même temps que votre exécutable lorsque vous donnerez votre programme à d'autres personnes. Sous Windows, il s'agit tout simplement d'un fichier SDL.dll. Celui-ci devra se trouver :
 - soit dans le même dossier que l'exécutable (ce que je recommande). L'idéal est de toujours donner la DLL avec votre exécutable et de la laisser dans le même dossier. Si vous placez la DLL dans le dossier de Windows, vous n'aurez plus besoin de joindre une DLL dans chaque dossier contenant un programme SDL. Toutefois, cela peut poser des problèmes de conflits de version si vous écrasez une DLL plus récente ;
 - soit dans le dossier `c:\Windows` .
 - **Development libraries** : ce sont les fichiers `.a` (ou `.lib` sous Visual) et `.h` vous permettant de créer des programmes SDL. Ces fichiers ne sont nécessaires que pour vous, le programmeur. Vous n'aurez donc pas à les distribuer avec votre programme une fois qu'il sera fini.

Si vous êtes sous Windows, on vous propose trois versions dépendant de votre compilateur :

- - VC6 : pour ceux qui utilisent Visual Studio payant dans une vieille version (ce qui a peu de chances de vous concerner) ; vous y trouverez des fichiers `.lib` ;
 - VC8 : pour ceux qui utilisent Visual Studio 2005 Express ou ultérieur ; vous y trouverez des fichiers `.lib` ;
 - mingw32 : pour ceux qui utilisent Code::Blocks (il y aura donc des fichiers `.a`).

La particularité, c'est que les « Development libraries » contiennent tout ce qu'il faut : les `.h` et `.a` (ou `.lib`) bien sûr, mais aussi la `SDL.dll` à distribuer avec votre application ainsi que la documentation de la SDL !

Bref, tout ce que vous avez à faire au final est de télécharger les « Development libraries ». Tout ce dont vous avez besoin s'y trouve.

Ne vous trompez pas de lien ! Prenez bien la SDL dans la section « Development libraries » et non le code source de la section « Source code » !

Qu'est-ce que la documentation ?

Une documentation, c'est la liste complète des fonctions d'une bibliothèque. Toutes les documentations sont écrites en anglais (oui, même les bibliothèques écrites par des Français ont leur documentation en anglais). Voilà une raison de plus pour progresser dans la langue de Shakespeare !

La documentation n'est pas un cours, elle est en général assez austère. L'avantage par rapport à un cours, c'est qu'elle est complète. Elle contient la liste de *toutes* les fonctions, c'est donc LA référence du programmeur.

Bien souvent, vous rencontrerez des bibliothèques pour lesquelles il n'y a pas de cours. Vous aurez uniquement la « doc' » comme on l'appelle, et vous devrez être capables de vous débrouiller avec seulement ça (même si parfois c'est un peu dur de démarrer sans aide). Un vrai bon programmeur peut donc découvrir le fonctionnement d'une bibliothèque uniquement en lisant sa doc'.

A priori, vous n'aurez pas besoin de la doc' de la SDL de suite car je vais moi-même vous expliquer comment elle fonctionne. Toutefois, c'est comme pour la bibliothèque standard : je ne pourrai pas vous parler de toutes les fonctions. Vous aurez donc certainement besoin de lire la doc' plus tard.

La documentation se trouve déjà dans le package « Development libraries », mais si vous le voulez vous pouvez la télécharger à part en vous rendant dans le menu **Documentation** / **Downloadable** . Je vous recommande de placer les fichiers HTML de la documentation dans un dossier spécial (intitulé par exemple **Doc SDL**) et de faire un raccourci vers le sommaire **index.html** . Le but est que vous puissiez accéder rapidement à la documentation lorsque vous en avez besoin.

Créer un projet SDL : Windows



L'installation d'une bibliothèque est en général un petit peu plus compliquée que les installations dont vous avez l'habitude. Ici, il n'y a pas d'installateur automatique qui vous demande simplement de cliquer sur **Suivant - Suivant - Suivant - Terminer** .

En général, installer une bibliothèque est assez difficile pour un débutant. Pourtant, si ça peut vous remonter le moral, l'installation de la SDL est beaucoup plus simple que bien d'autres bibliothèques que j'ai eu l'occasion d'utiliser (en général on ne vous donne que le code source de la bibliothèque, et c'est à vous de la recompiler !).

En fait, le mot « installer » n'est peut-être pas celui qui convient le mieux. Nous n'allons rien installer du tout : nous voulons simplement arriver à créer un nouveau projet de type SDL avec notre IDE. Or, selon l'IDE que vous utilisez la manipulation sera un peu différente. Je vais présenter la manipulation pour chacun des IDE que je vous ai présentés au début du cours pour que tout le monde puisse suivre.

Je vais maintenant vous montrer comment créer un projet SDL sous chacun de ces trois IDE.

Création d'un projet SDL sous Code::Blocks

1/ Extraction des fichiers de la SDL

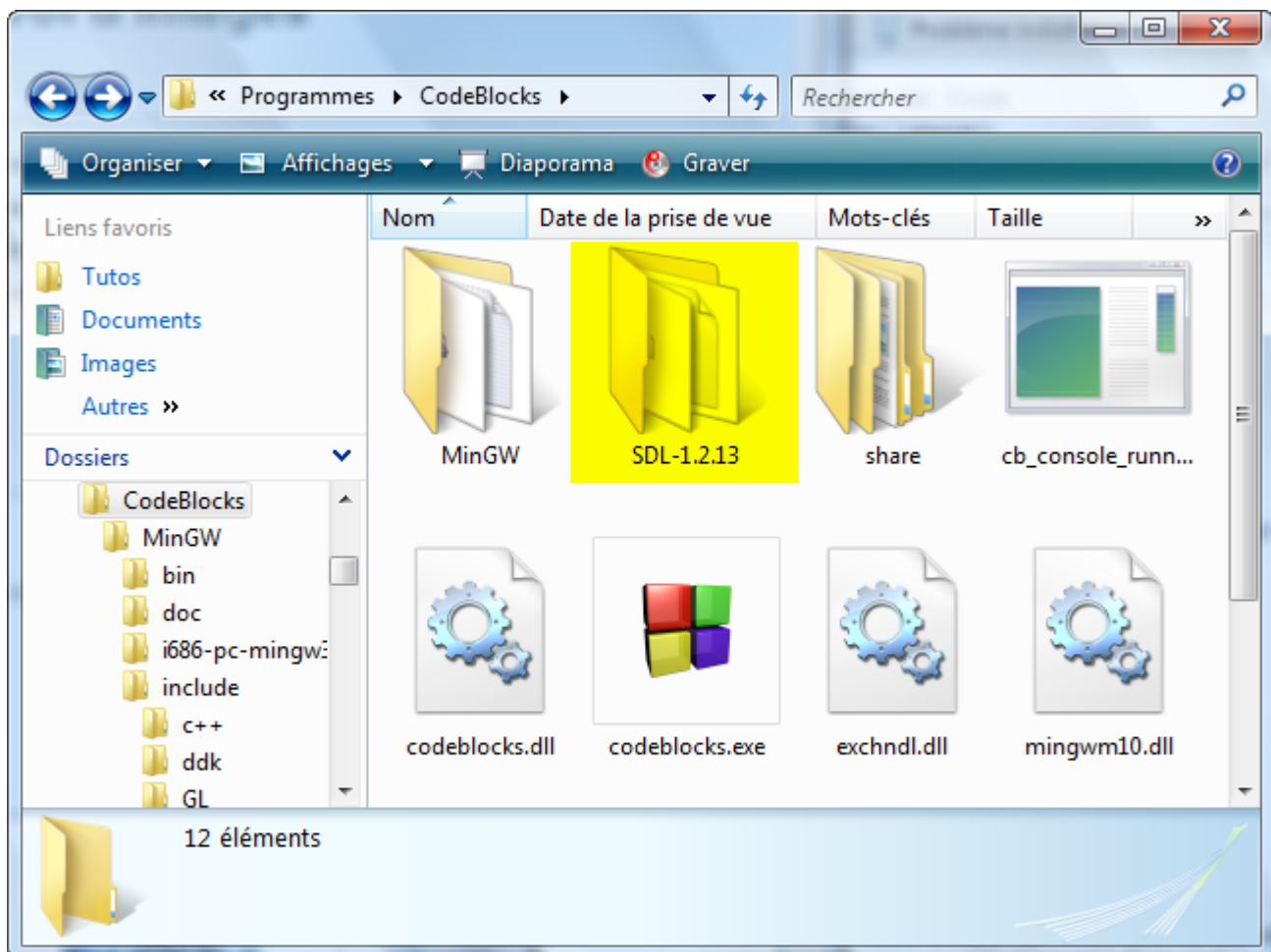
Ouvrez le fichier compressé de « Development Libraries » que vous avez téléchargé.

Ce fichier est un `.zip` pour Visual et un `.tar.gz` pour mingw32 (il vous faudra un logiciel comme Winrar ou 7-Zip pour décompresser le `.tar.gz`).

Le fichier compressé contient plusieurs sous-dossiers. Ceux qui nous intéressent sont les suivants :

- `bin` : contient la `.dll` de la SDL ;
- `docs` : contient la documentation de la SDL ;
- `include` : contient les `.h` ;
- `lib` : contient les `.a` (ou `.lib` pour Visual).

Vous devez extraire tous ces fichiers et dossiers quelque part sur votre disque dur. Vous pouvez par exemple les placer dans le dossier de Code::Blocks, dans un sous-dossier `SDL` (fig. suivante).



Dans mon cas, la SDL sera installée dans le dossier :


```
C:\Program Files\CodeBlocks\SDL-1.2.13
```

Retenez bien le nom du dossier dans lequel vous l'avez installée, vous allez en avoir besoin pour configurer Code::Blocks.

Maintenant, il va falloir faire une petite manipulation pour simplifier la suite. Allez dans le sous-dossier `include/SDL` (dans mon cas, il se trouve dans

```
C:\Program Files\CodeBlocks\SDL-1.2.13\include\SDL
```

). Vous devriez y voir de nombreux petits fichiers `.h` . Copiez-les dans le dossier parent, c'est-à-dire dans :

```
C:\Program Files\CodeBlocks\SDL-1.2.13\include
```

La SDL est installée ! Il faut maintenant configurer Code::Blocks.

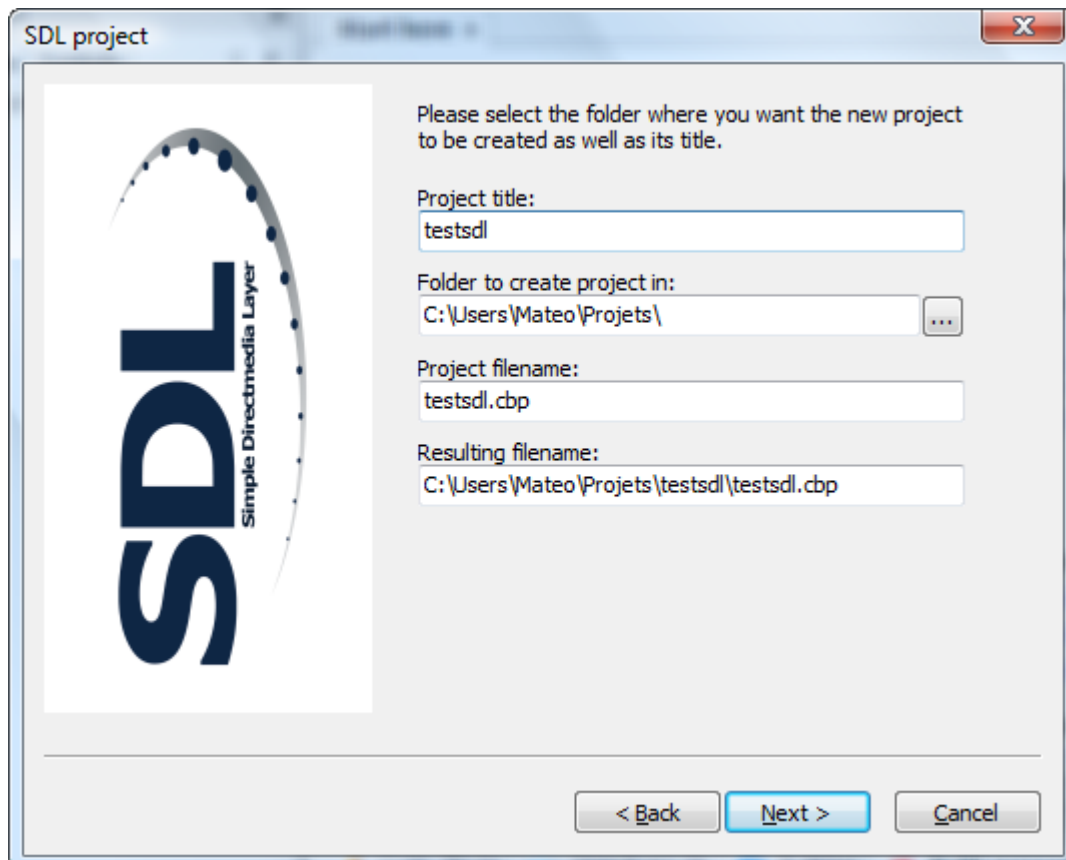
2/ Création du projet SDL

Ouvrez maintenant Code::Blocks et demandez à créer un nouveau projet.

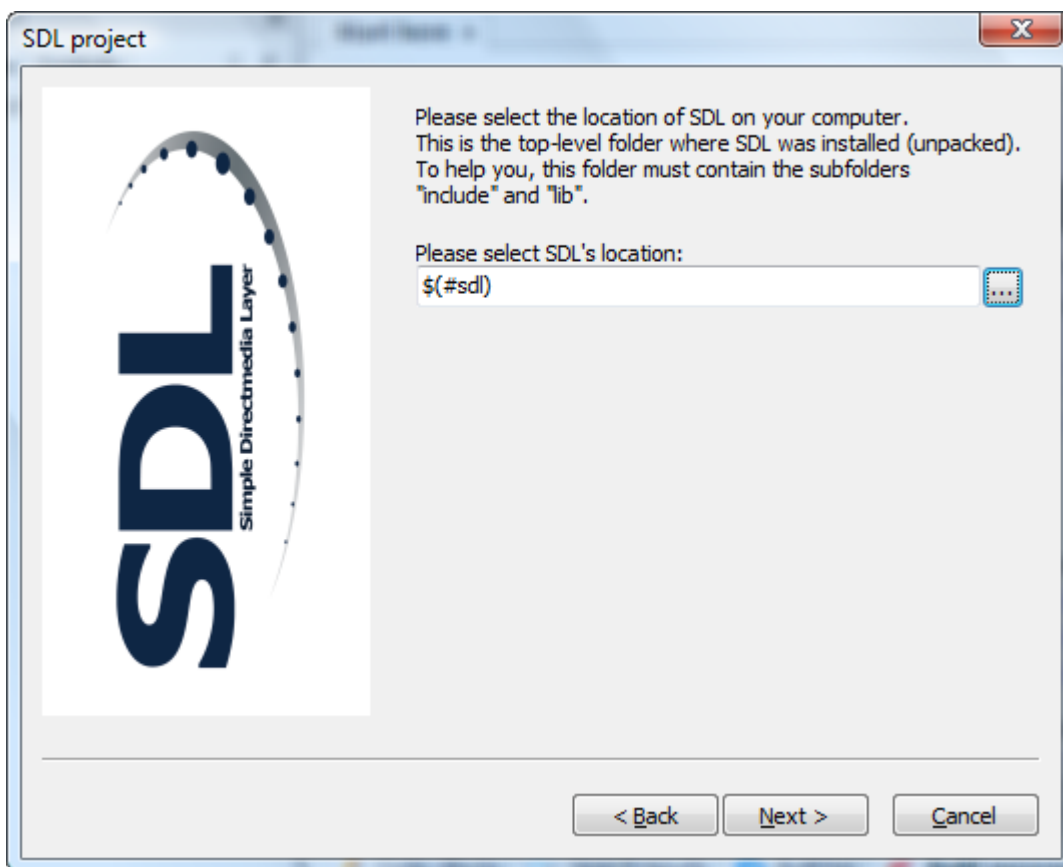
Au lieu de créer un projet `Console Application` comme vous aviez l'habitude de faire, vous allez demander à créer un projet de type `SDL project` .

La première fenêtre de l'assistant qui apparaît ne sert à rien, cliquez sur `Next` .

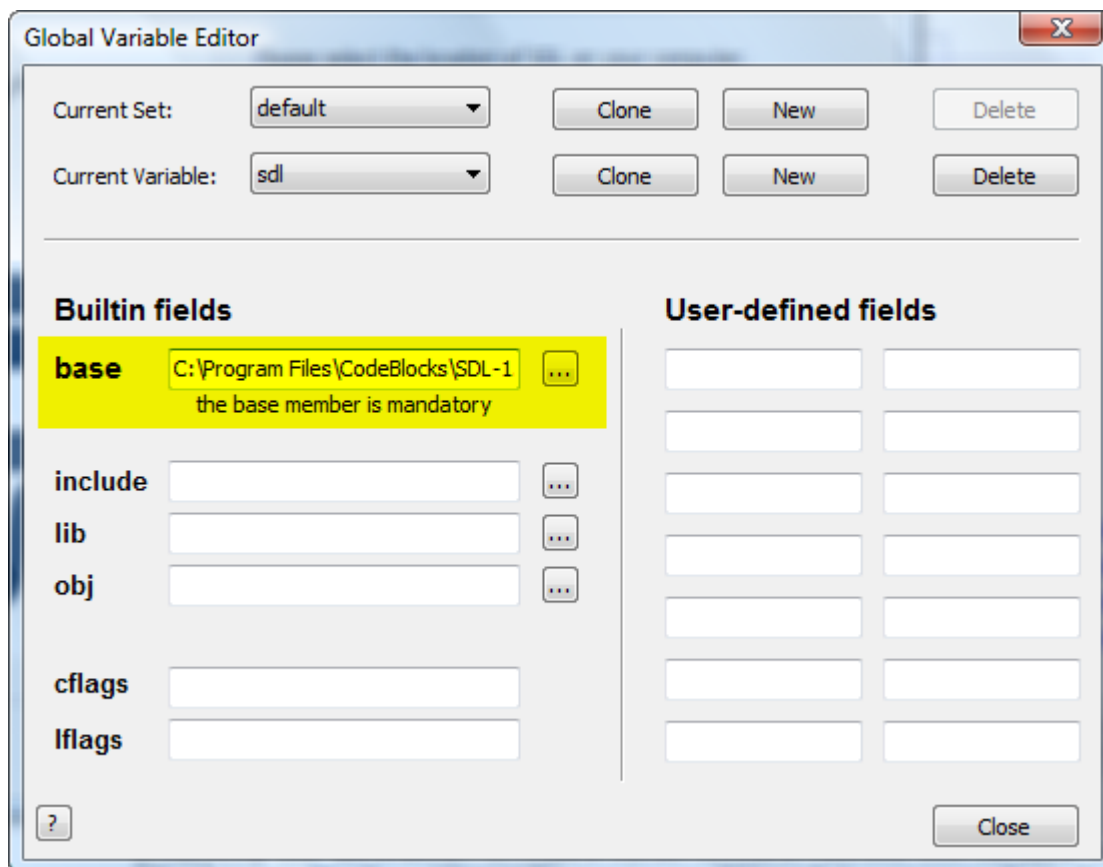
On vous demande ensuite le nom de votre projet et le dossier dans lequel il doit être placé, comme vous l'avez toujours fait (fig. suivante).



Vient ensuite la partie où vous devez indiquer où se trouve installée la SDL (fig. suivante).



Cliquez sur le bouton ... à droite. Une nouvelle fenêtre un peu complexe s'ouvre (fig. suivante).



Vous devez simplement remplir le champ nommé `base` . Indiquez le dossier où vous avez décompressé la SDL. Dans mon cas, c'est :

```
C:\Program Files\CodeBlocks\SDL-1.2.13
```

Cliquez sur `Close` . Une nouvelle fenêtre apparaît. C'est une fenêtre-piège (dont je n'ai toujours pas saisi l'intérêt). Elle vous demande un dossier. Cliquez sur `Annuler` pour ne rien faire.

Cliquez ensuite sur `Next` dans l'assistant, puis choisissez de compiler en mode `Release` ou `Debug` (peu importe) et enfin, choisissez `Finish` .

Code::Blocks va créer un petit projet SDL de test comprenant un `main.c` et un fichier `.bmp` . Avant d'essayer de le compiler, copiez la DLL de la SDL (vous devriez l'avoir copiée dans `C:\Program Files\CodeBlocks\SDL-1.2.13\bin\SDL.dll`) dans le dossier de votre projet.

Essayez ensuite de compiler : une fenêtre avec une image devrait s'afficher. Bravo, ça fonctionne !

Si on vous dit « Cette application n'a pas pu démarrer car SDL.dll est introuvable », c'est que vous n'avez pas copié le fichier `SDL.dll` dans le dossier de votre projet !

Il faudra penser à fournir cette `.dll` en plus de votre `.exe` à vos amis si vous voulez qu'ils puissent eux aussi exécuter le programme. En revanche, vous n'avez pas besoin de leur joindre les `.h` et `.a` qui n'intéressent que vous.

Vous pouvez supprimer le `.bmp` du programme, on n'en aura pas besoin.

Quant au fichier `main.c` , il est un peu long, on ne va pas démarrer avec ça. Supprimez tout son contenu et remplacez-le par :

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <SDL/SDL.h>
4
5 int main(int argc, char *argv[])
6 {
7
8     return 0;
9 }
```

C'est en fait un code de base très similaire à ceux que l'on connaît (un `include` de `stdlib` , un autre de `stdio` , un `main` ...).

La seule chose qui change, c'est le `include` d'un fichier `SDL.h` . C'est le fichier `.h` de base de la SDL qui se chargera d'inclure tous les autres fichiers `.h` de la SDL.

Création d'un projet SDL sous Visual C++

1/ Extraction des fichiers de la SDL

Sur le site de la SDL, téléchargez la dernière version de la SDL. Dans la section « Development Libraries », prenez la version pour Visual C++ 2005 Service Pack 1.

Ouvrez le fichier zip.

Il contient la doc' (dossier `docs`), les `.h` (dossier `include`), et les `.lib` (dossier `lib`) qui sont l'équivalent des `.a` pour le compilateur de Visual. Vous trouverez aussi le fichier `SDL.dll` dans ce dossier `lib` .

- - Copiez `SDL.dll` dans le dossier de votre projet.
 - Copiez les `.lib` dans le dossier `lib` de Visual C++. Par exemple chez moi il s'agit du dossier :

```
C:\Program Files\Microsoft Visual Studio 8\VC\lib .
```

- - Copiez les `.h` dans le dossier `includes` de Visual C++. Créez un dossier `SDL` dans ce dossier `includes` pour regrouper les `.h` de la `SDL` entre eux.

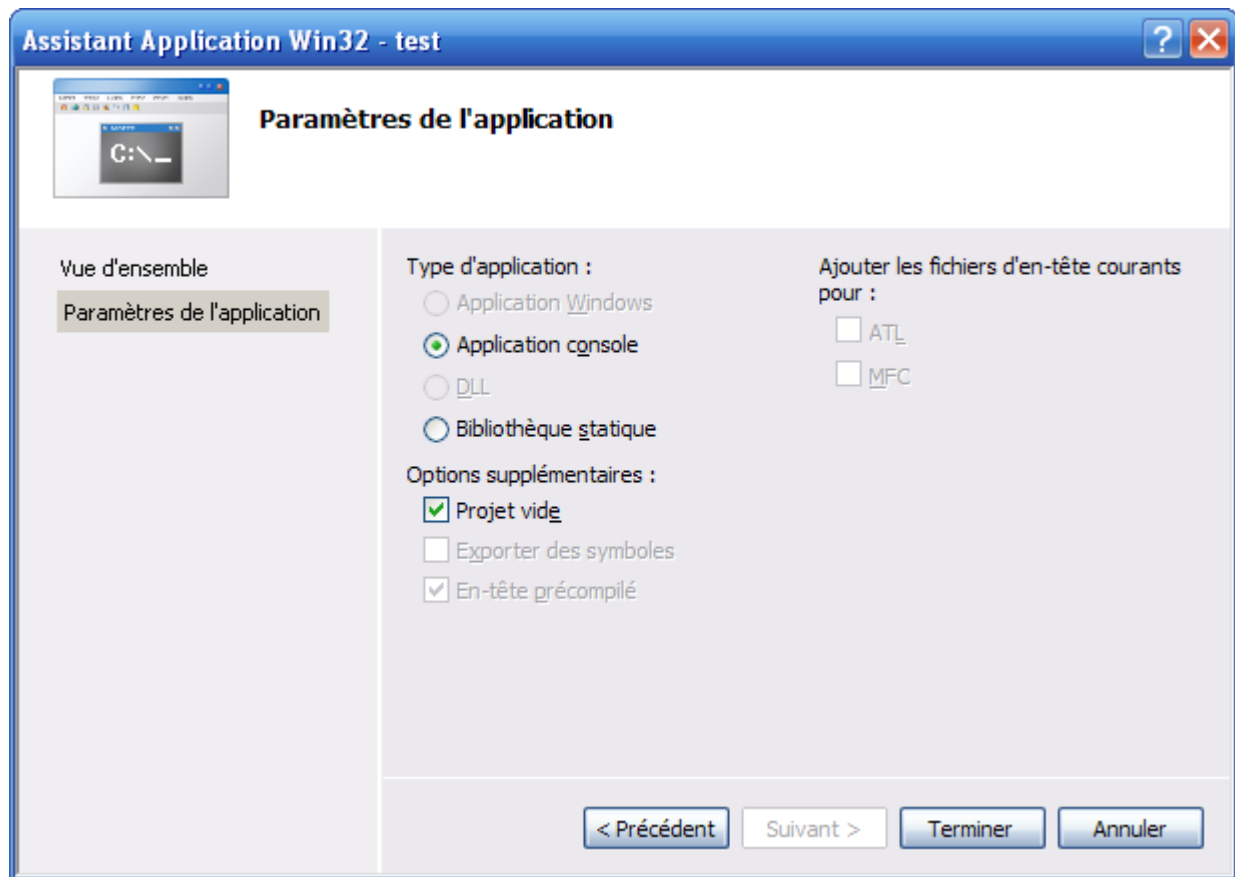
Chez moi, je mets donc les `.h` dans :

```
C:\Program Files\Microsoft Visual Studio 8\VC\include\SDL .
```

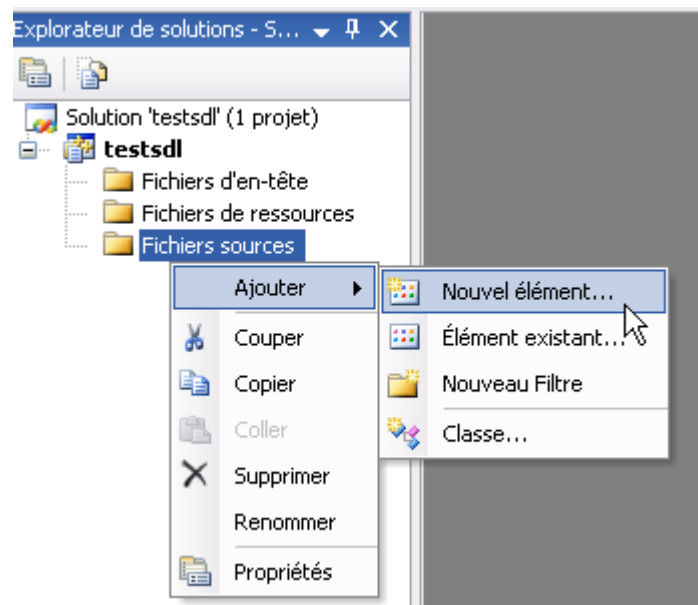
2/ Création d'un nouveau projet SDL

Sous Visual C++, créez un nouveau projet de type `Application console Win32` . Appelez votre projet `testsd1` par exemple. Cliquez sur OK.

Un assistant va s'ouvrir. Allez dans `Paramètres de l'application` et vérifiez que `Projet vide` est coché (fig. suivante).



Le projet est alors créé. Il est vide. Ajoutez-y un nouveau fichier en faisant un clic droit sur `Fichiers sources` , `Ajouter` / `Nouvel élément` (fig. suivante).



Dans la fenêtre qui s'ouvre, demandez à créer un nouveau fichier de type `Fichier C++ (.cpp)` que vous appellerez `main.c` . En utilisant l'extension `.c` dans le nom du fichier, Visual créera un fichier `.c` et non un fichier `.cpp` .

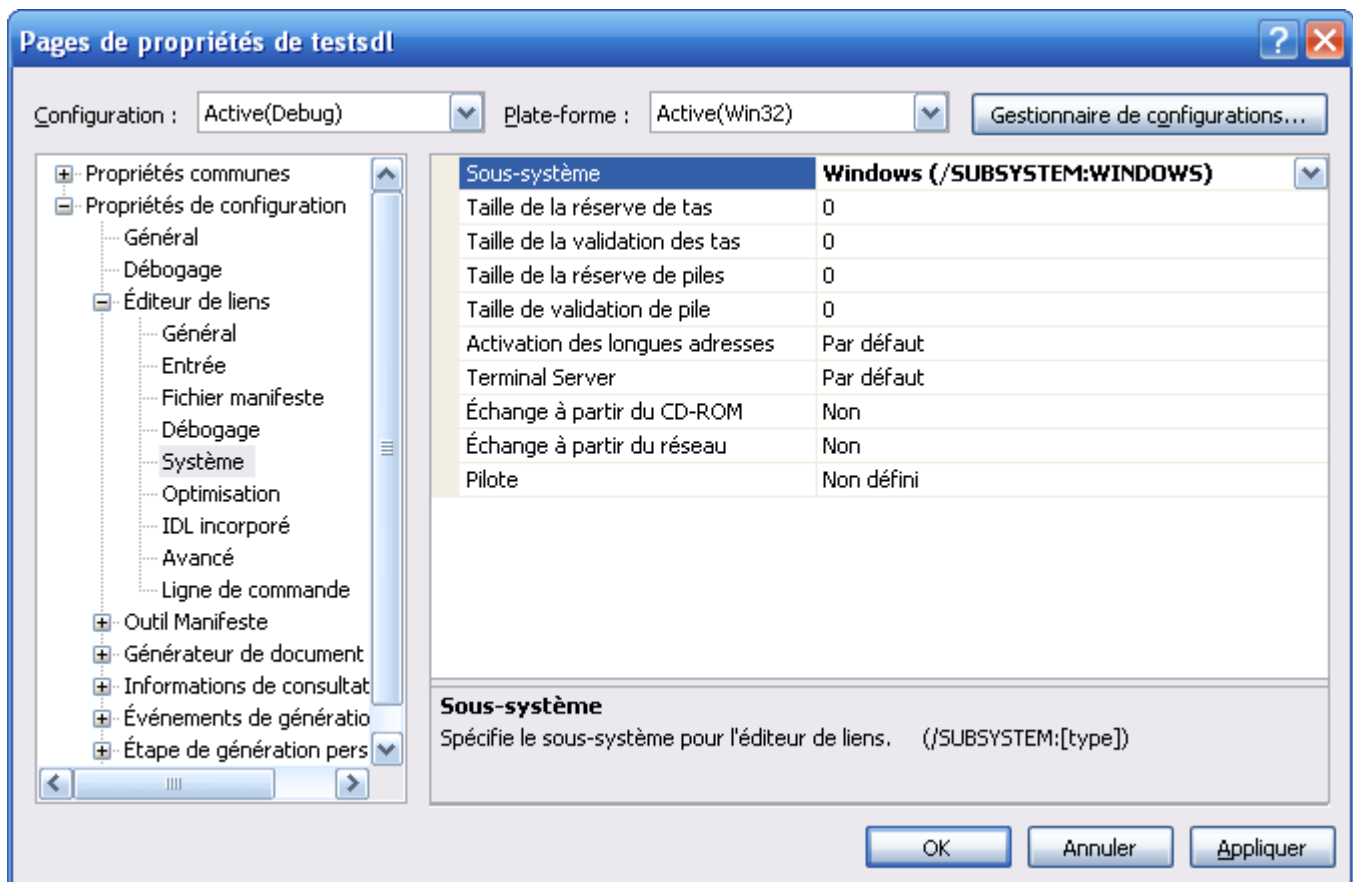
Écrivez (ou copiez-collez) le code « de base » mentionné précédemment dans votre nouveau fichier vide.

3/ Configuration du projet SDL sous Visual C++

La configuration du projet est un peu plus délicate que pour Code::Blocks, mais elle reste humainement faisable. Allez dans les propriétés de votre projet : **Projet** /

Propriétés de testsd1 .

- Dans la section **C / C++ => Génération de code** , mettez le champ **Bibliothèque runtime** à **DLL multithread (/MD)** .
- Dans la section **C/C++ => Avancé** , sélectionnez **Compilation sous** et optez pour la valeur **Compiler comme code C (/TC)** (sinon Visual vous compilera votre projet comme étant du C++).
- Dans la section **Éditeur de liens => Entrée** , modifiez la valeur de **Dépendances supplémentaires** pour y ajouter **SDL.lib SDLmain.lib** .
- Dans la section **Éditeur de liens => Système** , modifiez la valeur de **Sous-système** et mettez-la à **Windows** (fig. suivante).



Validez ensuite vos modifications en cliquant sur **OK** et enregistrez le tout.

Vous pouvez maintenant compiler en allant dans le menu **Générer / Générer la solution** .

Rendez-vous dans le dossier de votre projet pour y trouver votre exécutable (il sera peut-être dans un sous-dossier `Debug`). N'oubliez pas que le fichier `SDL.dll` doit se trouver dans le même dossier que l'exécutable. Double-cliquez sur votre `.exe` : si tout va bien, il ne devrait rien se passer. Sinon, s'il y a une erreur c'est probablement que le fichier `SDL.dll` ne se trouve pas dans le même dossier.

Créer un projet SDL : Mac OS (Xcode)



Cette section a été à l'origine rédigée par Pouet_forever du Site du Zéro, que je remercie à nouveau au passage.

Tout d'abord, rendez-vous sur le site de la SDL pour télécharger la version 1.2 pour Mac OS. Dans l'arborescence de gauche, cliquez sur `SDL 1.2` dans la partie `Download` , comme indiqué à la figure suivante.



En bas de la page, vous trouverez une partie `Runtime Libraries` . Téléchargez le fichier concernant votre architecture (Intel ou PowerPC), comme sur la figure suivante. Pour connaître votre architecture, il suffit d'aller dans le menu `Pomme` en haut à gauche et de cliquer sur `À propos de ce Mac` . Sur la ligne `Processeur` , vous verrez soit Intel, soit PowerPC.

Source Code:

[SDL-1.2.15.tar.gz](#) - [GPG signed](#)
[SDL-1.2.15-1.src.rpm](#) - [GPG signed](#)
[SDL-1.2.15.zip](#) - [GPG signed](#)

Runtime Libraries:**Linux:**

[SDL-1.2.15-1.i386.rpm](#)
[SDL-1.2.15-1.x86_64.rpm](#)

Win32:

[SDL-1.2.15-win32.zip](#)
[SDL-1.2.15-win32-x64.zip](#) (64-bit Windows)

Mac OS X:

[SDL-1.2.15.dmg](#) (Intel 10.5+)
[SDL-1.2.15-OSX10.4.dmg](#) (Intel/PPC 10.4)

Development Libraries:**Linux:**

[SDL-devel-1.2.15-1.i386.rpm](#)
[SDL-devel-1.2.15-1.x86_64.rpm](#)

Win32:

[SDL-devel-1.2.15-VC.zip](#) (Visual C++)
[SDL-devel-1.2.15-mingw32.tar.gz](#) (Mingw32)

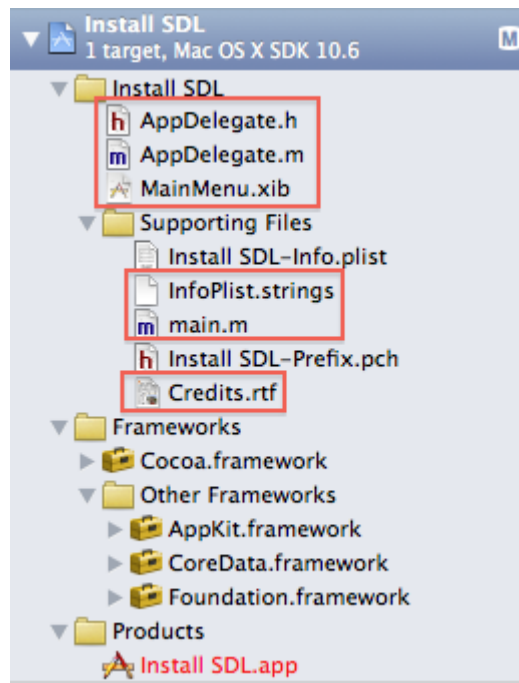
Une fois le fichier téléchargé, montez-le (double-cliquez dessus) ; il devrait alors s'ouvrir tout seul. Dans ce dossier, vous trouverez un dossier `SDL.framework`. Copiez-le et collez-le dans le dossier `/Library/Frameworks` (en français `/Bibliothèque/Frameworks`).

Ça y est, notre bibliothèque est installée !

Vous trouverez un autre dossier qui se nomme `devel-lite` ; gardez-le ouvert, nous nous en servons par la suite.

Maintenant, créez un nouveau projet Cocoa Application, puis cliquez sur `Next`. Dans `Product Name`, nommez votre projet (« SDL » par exemple) et dans `Company Identifier`, mettez ce que vous voulez (votre pseudo par exemple). Laissez le reste tel quel puis cliquez sur `Next`. Choisissez où vous souhaitez mettre votre projet. Un dossier sera créé, donc pas besoin d'en créer un et d'y placer votre projet.

Une fois votre projet créé, vous pouvez supprimer tous les fichiers qui ne nous serviront pas : `AppDelegate.h`, `AppDelegate.m`, `MainMenu.xib`, `InfoPlist.strings`, `main.m` et `Credits.rtf` (figure suivante).

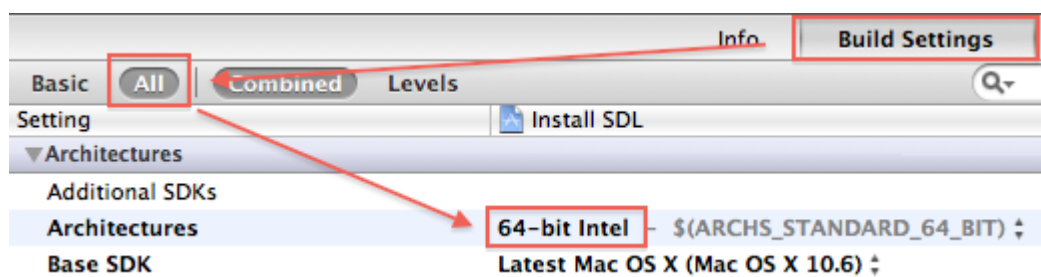


Sélectionnez votre projet dans l'arborescence de gauche (**Install SDL** sur la figure suivante). Dans la 2^e arborescence, sélectionnez le nom de votre projet sous **PROJECT** (et pas **TARGETS**) (figure suivante).

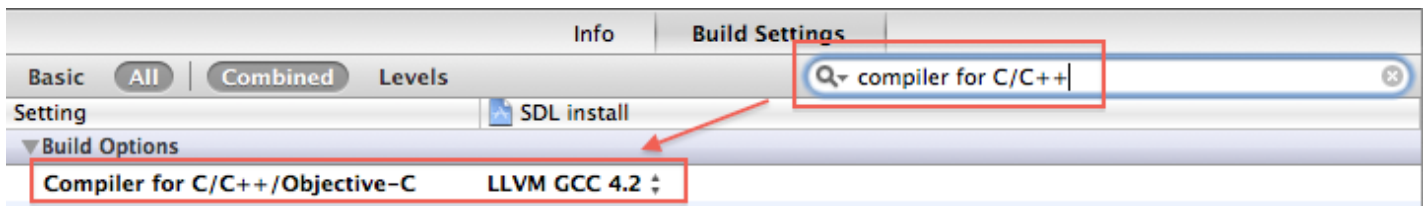


Vous pouvez éventuellement changer la localisation de **English** à **French** . Sélectionnez **English** , cliquez sur le **-** pour le supprimer et cliquez sur le **+** pour ajouter **French** (si vous ne le faites pas, ça n'aura aucune incidence).

On va maintenant configurer notre projet en 32bits (la SDL ne fonctionnant pas en 64 bits) et ajouter les chemins pour les frameworks ainsi que les différents headers. Cliquez sur l'onglet **Build Settings** , puis **All** . Ensuite dans **Architectures** , cliquez sur **64-bit Intel** et sélectionnez **32-bit Intel** , comme à la figure suivante.



Une fois que c'est fait, sélectionnez LLVM GCC 4.2 sur la ligne Compiler for C/C++/Objective-C.



Allez dans le champ de recherche en haut à droite et tapez « search paths » ; vous devriez trouver les 2 lignes qui nous intéressent : `Header search paths` et `Framework search paths` . Double cliquez sur la zone à droite de la ligne `Framework search paths` , cliquez sur le `+` et ajoutez le chemin `/Library/Frameworks` . Pour `Header Search paths` ajoutez le chemin `/Library/Frameworks/SDL.framework/Headers` . Vous devriez avoir le même résultat qu'à la figure suivante.

Framework Search Paths	<code>/Library/Frameworks</code>
Header Search Paths	<code>/Library/Frameworks/SDL.framework/Headers</code>

Sélectionnez maintenant votre « cible » (en dessous de `TARGETS` cette fois-ci), comme à la figure suivante.

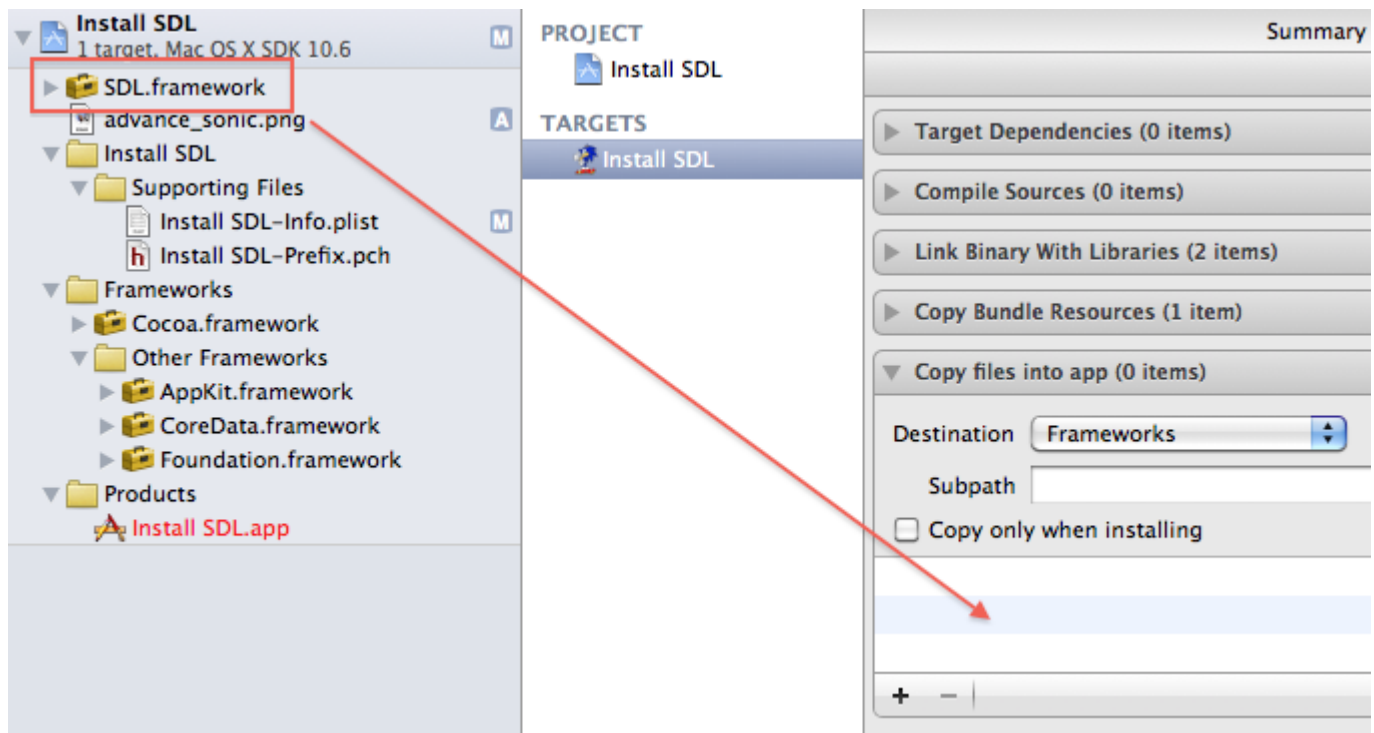


Allez dans l'onglet `Summary` . Dans `Application Category` , vous pouvez mettre ce que vous voulez, mais ça n'apporte pas grand chose, c'est pour l'AppStore. Modifiez la ligne `Main Interface` et mettez « `SDLMain` ». Le cadre `App Icon` , comme son nom l'indique, est pour définir une icône à votre programme. Il suffit de faire glisser l'image souhaitée dans le cadre (avec un *drag & drop*). Dans la partie `Linked Frameworks and Libraries` , on va ajouter notre framework : `SDL.framework` . Cliquez sur le `+` et dans le champ de recherche, tapez « `SDL` ». Quand vous avez repéré le framework , cliquez sur « `Add` ». Si vous ne trouvez pas le framework dans la liste déroulante, c'est que vous ne l'avez pas placé dans le bon dossier (`/Library/Frameworks` ou `/Bibliothèque/Frameworks` en français).

Les icônes sous Mac OS sont de type `.icns` . Si vous mettez un autre format, vous remarquerez que votre icône n'apparaît pas. Il faut donc que vous utilisiez une icône au format `.icns` . Si vous voulez convertir une image en icône, vous pouvez utiliser le logiciel **Icon Composer** qui se situe dans le dossier `/Developer/Applications/Utilities` . Il vous suffit simplement de glisser votre image dans le cadre souhaité et d'enregistrer votre image.

Dans l'onglet **Info**, on peut indiquer beaucoup d'informations concernant le programme ; je ne m'attarderai pas dessus, si vous voulez des informations, je vous conseille de lire la doc d'Apple à ce sujet. Les deux seules choses que vous pouvez éventuellement modifier sont la **Localization**, de **en** en **fr**, ainsi que le **Copyright** pour y mettre ce que vous voulez (le **__MyCompanyName__** n'est pas génial).

Allez ensuite dans l'onglet **Build Phases** et cliquez sur **Add Build Phase > Copy Files** en bas à droite de la fenêtre. Double cliquez sur le nom **Copy Files** et renommez le en « **Copy frameworks into app** ». Dans **Destination**, sélectionnez **Frameworks**. Pour ajouter vos frameworks, glissez-les de l'arborescence de gauche dans votre **Build phase**, comme à la figure suivante.



Je vous conseille de ranger tous vos frameworks dans le dossier **Framework**. Cela vous permettra de vous y retrouver plus facilement.

De même pour les fichiers sources, je vous conseille de créer des dossiers pour les ranger correctement. Pour créer un dossier, il faut faire un clic droit et **New Group**, ensuite vous glissez vos fichiers dedans.

Nous allons maintenant ajouter les fichiers **SDLMain.h** et **SDLMain.m**. Allez dans le dossier **devel-lite** ouvert précédemment et ajoutez les 2 fichiers au projet. Si une fenêtre vous demandant de choisir des options pour la copie s'affiche, cochez la case **Copy items into destination group's folder (if needed)**.

Dernière ligne droite : créez un fichier **main.c**. Allez dans le menu **File > New > New File**, puis dans **C and C++**. Sélectionnez **C File** puis **Next**. Nommez votre fichier et le tour est joué !

Créer un projet SDL : Linux



Si vous compilez sous Linux avec un IDE, il faudra modifier les propriétés du projet (la manipulation sera quasiment la même). Si vous utilisez Code::Blocks (qui existe aussi en version Linux) vous pouvez suivre la même procédure que celle décrite dans la partie Windows.

Et pour ceux qui compilent à la main ?

Il y en a peut-être parmi vous qui ont pris l'habitude de compiler à la main sous Linux à l'aide d'un *Makefile* (fichier commandant la compilation).

Si c'est votre cas, je vous invite à télécharger un *Makefile* que vous pouvez utiliser pour compiler des projets SDL.

[Télécharger le Makefile](#)

La seule chose un peu particulière, c'est l'ajout de la bibliothèque SDL pour le linker (`LDFLAGS`). Il faudra que vous ayez téléchargé la SDL version Linux et que vous l'ayez installée dans le dossier de votre compilateur, de la même manière qu'on le fait sous Windows (dossiers `include/SDL` et `lib`)

Ensuite, vous pourrez utiliser les commandes suivantes dans la console :

```
make          #Pour compiler le projet
make clean    #Pour effacer les fichiers de compilation (.o inutiles)
make mrproper #Pour tout supprimer sauf les fichiers source
```

En résumé

- La SDL est une bibliothèque de bas niveau qui permet d'ouvrir des fenêtres et d'y faire des manipulations graphiques en 2D.
- Elle n'est pas installée par défaut, il faut la télécharger et configurer votre IDE pour l'utiliser.
- Elle est libre et gratuite, ce qui vous permet de l'utiliser rapidement et vous garantit sa pérennité.
- Il existe des milliers d'autres bibliothèques dont beaucoup sont de très bonne qualité. C'est la SDL qui a été sélectionnée pour la suite de ce cours pour son aspect ludique. Si vous souhaitez développer des interfaces complètes avec menus et boutons par la suite, je vous invite à vous pencher sur la bibliothèque GTK+ par exemple.



QUIZ : QUIZ 2

CRÉATION D'UNE FENÊTRE ET DE SURFACES



Le professeur

Mathieu Nebra

Entrepreneur à plein temps, auteur à plein temps et co-fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...



Livre



PDF

OPENCCLASSROOMS



ENTREPRISES



CONTACT



EN PLUS



 Français



