**Questions SDL**

1. la SDL 2 est une mise a jour majeur de  Simple DirectMedia Layer (SDL) qui est une bibliothèque logicielle libre.
2. les fonction majeur de la SDL2 sont :
   1. accélération matérielle 3D complète
   2. support d'OpenGL 3.0 et supérieur avec de nombreux profils (core, compatibility, debug, robust, etc.)
   3. support d'OpenGL ES
   4. support de multiples fenêtres
   5. support de multiples moniteurs
   6. support de multiples périphériques audio
   7. support d'Android et iOS
   8. bibliothèque de rendus simple 2D pouvant utiliser Direct3D, OpenGL, OpenGL ES ou un rendu logiciel
   9. retour de force disponible sur Windows, Mac OS X et Linux
   10. support de XInput et XAudio 2 pour Windows
   11. opérations atomiques
   12. gestion de l'énergie (indique l'autonomie de la batterie, etc.)
   13. fenêtre aux formes personnalisées
   14. audio 32 bits (entier et virgule flottante)
   15. bibliothèque simplifiée pour les manettes de jeu (la bibliothèque Joystick est toujours là !)
   16. support du toucher (multi-touch, mouvements, etc.)
   17. meilleur support du mode plein écran
   18. meilleur support du clavier (scancodes VS keycodes, etc.)
   19. boîtes de messages
   20. support du presse-papier
   21. support basique du glisser/déposer
   22. support correct des entrées Unicode et IME
   23. licence zlib à la place de la LGPL
   24. de nombreux ennuis ont disparu depuis la SDL 1.2
3. la SLD2 est disponible sur Windows, Mac OS X et Linux.
4. Et 6) Il faut télécharger les fichiers sdl2.

Ensuite il faut mettre le dossier SDL2-2.0.3 **à la racine** du dossier Code::Blocks.

Maintenant, on va rajouter des fichiers au dossier de votre choix (***i686*** ou ***x86\_x64***), afin qu'on puisse:

* rajouter des images **autre** que le format BMP dans votre interface graphique (***SDL\_image***)
* rajouter du texte dans votre interface graphique (***SDL\_ttf***)
* rajouter de la musique dans votre programme (***SDL\_mixer***)

Pour cela, il faut fusionner les dossier de SDL\_(***image***, ***ttf*** ou ***mixer***) avec les dossiers de **SDL2-2.0.3** (***i686*** ou ***x86\_64***):

* le dossier **bin**
* le dossier **lib**
* le dossier **include**
* le dossier **pkgconfig** à l'intérieur du dossier lib

Puis copier le contenu des dossiers **bin** et **"include\SDL2\"** du dossier SDL\_image dans ceux du dossier **"SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix"** dans Code::Blocks :

Ensuite, copier le contenu des dossiers **lib** et **pkgconfig** du dossier ***SDL\_image*** dans ceux du dossier **"SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix"** dans Code::Blocks :

Il faut ensuite faire la même chose pour ***SDL\_ttf*** et ***SDL\_mixer***.

Configuration sur code blocks :

Maintenant, allez sur "settings\compiler" et une fenêtre va apparaître:

Afin que Code::Blocks ne retourne pas d'erreur lié à "mcount", cochez la deuxième case **"Profile code when executed [-pg]"**.

Maintenant, allez sur l'onglet "Linker settings" afin d'indiquer le chemin des fichiers lib. Les fichiers lib important pour le développement avec SDL.

**ATTENTION**: l'ordre des linkers est ultra important, les étapes qui vont suivre doivent être respectés à la lettre.

Dans "link librairies", ajoutez les éléments suivants en provenance du dossier Code::Blocks:

* \MinGW\lib\libmingw32.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2.dll.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2\_image.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2\_image.dll.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2\_mixer.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2\_mixer.dll.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2\_ttf.a
* \SDL2-2.0.3\dossier\_architecture\_de\_votre\_choix\lib\libSDL2\_ttf.dll.a

Ensuite, rajoutez dans "Other linker", les linkers suivants afin d'éviter que le compilateur ne retourne une erreur liée aux fichiers lib non trouvés:

* -lmingw32
* -lSDL2main
* -lSDL2
* -lSDL2\_image
* -lSDL2\_mixer
* -lSDL2\_ttf

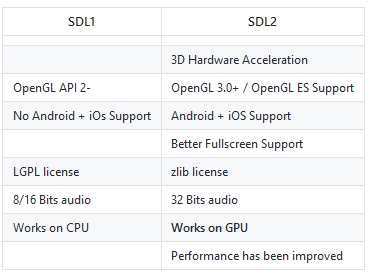
Enfin, dans "linker", ajoutez le chemin menant vers le dossier lib de votre librairie SDL 2.

Maintenant, allez sur l'onglet *"Search directories"* pour indiquer au compilateur où sont le dossier include et dossier lib pour SDL2.

Dans *"compiler"*, ajoutez le chemin menant vers le dossier include de votre librairie SDL 2.

Votre installation est prête.

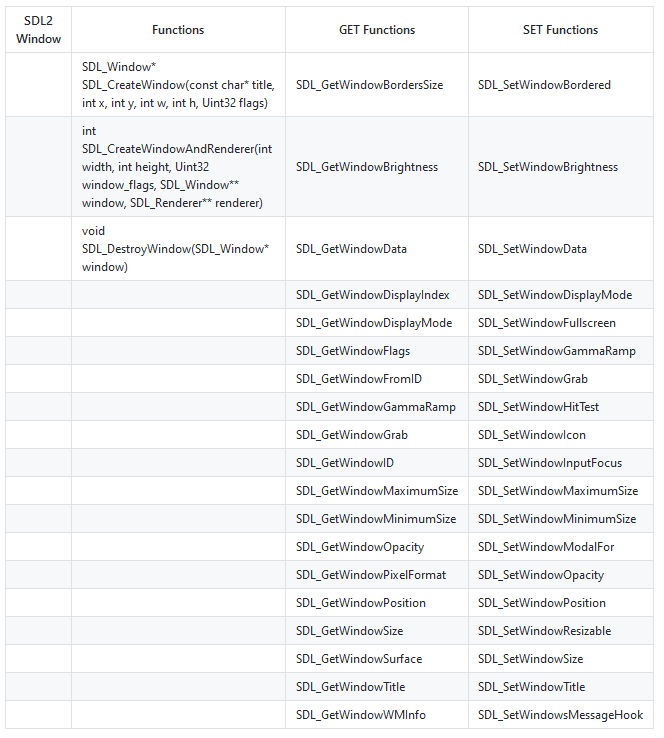
1. Différence SDL1 et SDL2 :



* 1. pour ouvrir une fenêtre en SDL 2, il suffit de taper le code: " **SDL\_Window\* screen = SDL\_CreateWindow("My window LUDUS", SDL\_WINDOWPOS\_UNDEFINED, SDL\_WINDOWPOS\_UNDEFINED, 480, 170, NULL);**"

(Code joint en annexe)

* 1. Primitives associées à la fenêtre :



9) le renderer est contenu dans une structure appelée **SDL\_Renderer**. Elle contient les informations du contexte d'affichage de la fenêtre

10) SDL\_Rect srcrect; SDL\_Rect défini une zone rectangulaire en x, y , h , w

SDL\_Rect est type de la bibliothèque SDL.

​SDL\_Point window\_position = {        *//    Position of windo*w

       ​SDL\_WINDOWPOS\_CENTERED,

       ​SDL\_WINDOWPOS\_CENTERED

   ​};

SDL\_Point est type structuré comme SDL\_Rect dans la bibliothèque SDL, on utilise un tableau 2D pour définir les points.

11)     SDL\_SetRenderDrawColor(renderer, 255, 0, 0, 255); //Change la couleur du background

12) code joint en annexe.

13) code joint en annexe

14) La fonction la plus utilisée permettant de dessiner des points et des lignes est :

SDL\_RenderDrawLine (pour les lignes) et SDL\_RenderDrawPoint (pour les points)

15) Le SDL\_RenderClear sert à supprimer l’ancien render, c’est-à-dire l’ancien emplacement du render.

Le SDL\_RenderPresent sert à « faire passer au premier plan » ce que l’on a dessiner.

16) SDL\_Delay sert à instaurer un timer pour une action donnée. Souvent on va l’utiliser pour laisser afficher quelque chose un certain temps. Par exemple SDL\_Dedlay(5000) va laisser la fenêtre afficher pendant 5 secondes (car SDL\_Delay est exprimé en millisecondes).

17) Les surfaces en SDL2 sont l’équivalent des textures en SDL1. Ce sont les « endroits » ou l’on va dessiner, produire quelque chose.

18) Pour créer une surface, on doit utiliser la variable SDL\_Surface et comme la première surface que l’on doit créer est toujours la surface de l’écran, voici le code de la surface de l’écran :

SDL\_Surface \*ecran = NULL;

19) La fonction qui permet de colorer une surface avec une couleur unie s'appelle SDL\_FillRect(*FillRect* = « remplir rectangle » en anglais). Elle prend trois paramètres, dans l'ordre :

* un pointeur sur la surface dans laquelle on doit dessiner (par exemple écran) ;
* la partie de la surface qui doit être remplie. Si vous voulez remplir toute la surface (et c'est ce qu'on veut faire), envoyez NULL;
* la couleur à utiliser pour remplir la surface

Ce qui donne le code suivant : SDL\_FillRect (surface, NULL, couleur);

20) SDL\_BlitSurface() permet de copier une surface sur une autre.

Code d’exemple :

SDL\_BlitSurface(surface, &source\_rect, temp\_surface, NULL);

21)

Une texture est une structure SDL\_Texture. Nous avons dit qu’on pouvait le voir comme un paquet de pixels (un rectangle plus précisément). Ce paquet de pixels, on pourra l’afficher, le modifier, etc.

Creation d’une texture:

SDL\_Texture\* SDL\_CreateTexture (SDL\_Renderer\* renderer,

Uint32 format,

int access,

int w,

int h)

22) exemple de code on l’on va dessiner un carré sur une texture :

texture = SDL\_CreateTexture(renderer, SDL\_PIXELFORMAT\_RGBA8888,

SDL\_TEXTUREACCES\_TARGET, 200, 200);

SDL\_Rect rect = {50, 50, 100, 100};

SDL\_SetRenderDrawColor (renderer, 150, 0, 150, 255); /\* On dessine en violet \*/

SDL\_SetRenderTarget(renderer, texture); /\* On va dessiner sur la texture \*/

SDL\_RenderFillRect(renderer, &rect);

SDL\_SetRenderTarget(renderer, NULL);

23) Afficher une texture consiste à copier la texture sur le renderer puis à mettre à jour le renderer. Ainsi, on verra bien la texture à l’écran. La copie de la texture se fait avec la fonctionSDL\_RenderCopy dont le prototype est :

int SDL\_RenderCopy(SDL\_Renderer\* renderer,

SDL\_Texture\* texture,

const SDL\_Rect\* srcrect,

const SDL\_Rect\* dstrect)

24) La fonction **SDL\_QueryTexture** permet de récupérer plusieurs informations sur la texture, dont la largeur et la hauteur.

Exemple de code :

SDL\_Texture\* source;

int w, h;

SDL\_QueryTexture(source, NULL, NULL, &w, &h);

25) Code joint en annexe.

26) Création d’une texture à partir d’une image :

const char\* image1 = "Image1.bmp";

SDL\_Surface\* src\_surface = SDL\_LoadBMP(image1);

if(src\_surface == NULL){

printf("Image non reconnu");

}

SDL\_Texture\* text\_surface = SDL\_CreateTextureFromSurface(renderer, src\_surface);

27)

SDL\_SetRenderTarget(renderer, text\_surface);

SDL\_RenderCopy(renderer, text\_surface, NULL, NULL);

SDL\_RenderPresent(renderer);

28) Destruction des instances :

SDL\_DestroySurface(surface);

SDL\_DestroyTexture(texture);

SDL\_DestroyRenderer(renderer);

SDL\_DestroyWindow(window);

29)