Qu’est-ce que la SDL (2) ?

La SDL (Simple Directmedia Layer) est une librairie de fonctions, méthodes, de classes, etc prévue pour des applications et des logiciels 2D comme des jeux, des démos, etc.

A noter que la SDL dispose d’une licence zlib la rendant très permissive.

Quelles nouveautés dans la SDL 2 ?

Depuis la version SDL2 (2.0.0), l’accélération matérielle est possible, la licence a changé pour une licence Zlib au lieu de la LGPL et certains systèmes ne sont plus compatibles.

La verison suivante (2.0.6) ajoute un support multiplateforme de l'API graphique Vulkan, des modes de blends permettant la gestion de composition 2D, le support de nouveaux contrôles de jeu, une fonction de détection pour l’architecture ARM, une fonction de copie de surface, un support expérimental du pilote audio JACK, et des fonctions de rééchantillonnage sonore et de changement d'échelle graphique.

Que peut-elle faire ?

La SDL peut nativement gérer :

- L’affichage vidéo
- Les évènements
- L’audio numérique
- La gestion de périphérique
- Le multithreading
- Les timers

En ajoutant des modules, elle peut également gérer :

- Différentes polices d’écritures
- Plus de formats d’images et de sons
- Le dessin 2D
- Des fonctionnalités réseau
- La manipulation d’image

Et sur quoi peut-elle le faire ?

La SDL 2 fonctionne sur Windows, BeOS, Mac OS et la plupart des systèmes UNIX comme Mac OS X, Linux, Android, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, et QNX.

Installation de la SDL 2

Installation dans le cadre d'une utilisation sur Code :: Blocks avec MinGW

- Télécharger les différents prérequis.
  Code :: Blocks
  SDL 2 dans sa dernière version
  Les modules supplémentaires en fonction des besoins (SDL_image, SDL_ttf, SDL_mixer dans ce cas-ci)
- Préparer l'installation.
  Installez Code :: Blocks, décompressez les archives de la SDL et identifiez dans le dossier `SDL2-[2.x.x]` quelle version garder entre la 32bits et 64bits.
  Dans notre cas, il va falloir utiliser la 32bits de toutes façons, qui est dans le dossier `i686-w64-mingw32`, supprimez donc le dossier `x86_64-w64-mingw32`.
- Installer
  Une fois Code :: Blocks installé, copiez dans son dossier d'installation (là où se trouve `codeblocks.exe`) le dossier `SDL2-[2.x.x]`.
  Puis ajoutez vos modules supplémentaires en copiant les fichiers des dossiers `bin`, `lib`, `include` et `pkgconfig` de chacun d'eux dans les dossiers correspondants de `SDL2-[2.x.x]`.
  Gardez votre explorateur de fichiers ouvert, on en aura besoin.
- Configurer dans Code :: Blocks
  Ouvrez Code :: Blocks, allez dans `settings/compiler`. Une fenêtre apparait.
  Dans l'onglet *Compiler settings*, cochez la deuxième case "`Profile code when executed [-pg]`".
  Dans l'onglet *Linker settings*, ajoutez les chemins vers les fichiers lib dans l'encart de gauche `Link libraries`. L'ordre suivant est important !

```
\MinGW\lib\libmingw32.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2.dll.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2_image.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2_image.dll.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2_mixer.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2_mixer.dll.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2_ttf.a
\SDL2-2.0.3\i686-w64-mingw32\lib\libSDL2_ttf.dll.a
```

Il s'agit des chemins que vous devriez avoir mais dans le cas contraire, reportez vous à votre explorateur de fichiers pour les vérifier et les corriger.

Ajoutez également dans l'encart de droite, Other linker options, les liens suivants.

```
-lmingw32
-lSDL2main
-lSDL2
-lSDL2_image
-lSDL2_mixer
-lSDL2_ttf
```

Dans l'onglet *Search directories*, allez dans le sous-onglet *Compiler* pour indiquer l'emplacement du dossier `include` de votre SDL qui devrait ressembler à :

```
C:\Program Files (x86)\CodeBlocks\SDL2-2.0.10\i686-w64-
mingw32\include
```

Puis allez dans le sous-onglet *Linker* pour indiquer l'emplacement du dossier `lib` de votre SDL qui devrait ressembler à :

```
C:\Program Files (x86)\CodeBlocks\SDL2-2.0.10\i686-w64-mingw32\lib
```

(Pour avoir ces chemins complets, dans votre explorateur de fichiers, allez jusqu'au dossier concerné et copiez-collez le chemin directement depuis l'explorateur)

- Installation terminée !

Initialisation d'une instance SDL 2

```c
#include <SDL2/SDL.h>
//indique au compilateur qu'il va devoir inclure l'ensemble des fonctions, méthodes, etc du fichier SDL.h
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    return 0;
}
```

Pour la fonction `main`, il est nécessaire que sa signature soit complète pour le bon fonctionnement de la SDL, c'est-à-dire que l'on doit retrouver ses arguments dans sa syntaxe même s'ils ne sont pas utilisés.
Il n'est pas possible de se permettre un `int main()` ou encore `int main(void)`, et c'est pourquoi on doit avoir tous les arguments tel que `int main(int argc, char *argv[])` ou `int main(int argc, char **argv)`.

La surface

Vulgairement, en SDL, la surface est une zone « graphique » de mémoire dédié au dessin, et plus exactement au dessin pixel par pixel. La surface est opposée à la texture pour cela : elle est plus limitée que la texture en termes de possibilité mais permet une édition de chaque pixel plus facile.
(Code de création d'une surface dans « Création d'une fenêtre et de son rendu »)

À noter que l'on peut copier une surface (partielle ou non) vers une autre avec `SDL_BlitSurface()`.

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_BlitSurface(*surface1, NULL, *surface2, NULL);
    //[source, sélection source, cible, sélection cible]
    //Copie les pixels de surface1 vers surface2. En indiquant NULL pour les sélections, on copie la surface entière.
    …
    return 0;
}
```

La texture

La texture en SDL désigne un ensemble rectangulaire de pixel indépendant du renderer. Pour vulgariser, il s'agit plus ou moins d'un sprite que l'on va pouvoir manipuler avec la SDL sans avoir besoin de la redessiner constamment.

(Code de création d'une texture dans « Création d'une fenêtre et de son rendu »)

Dessiner dans une texture :

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_RenderTarget(*renderer, *texture);
    //[context, cible]
    //Permet de désigner une texture comme contexte pour dessiner
    …
    return 0;
}
```

Copier une texture :

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_RenderCopy(*renderer, *texture, NULL, NULL);
    //[context, cible, portion du contexte, sélection cible]
    //Permet de copier une texture vers un contexte. En indiquant NULL
    pour les sélections, on copie la texture entière.
    …
    return 0;
}
```

Interroger les attributs d'une texture :

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_QueryTexture(*texture, NULL, NULL, *w, *h);
    //[texture, format, chemin, largeur, hauteur]
    //Permet d'obtenir les attributs d'une texture. Indiquer NULL pour
    les éléments qui ne nous intéressent pas.
    …
    return 0;
}
```

Les images

Charger une image :

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_Surface* monImage = SDL_LoadBMP("images/monImage.bmp");
    //[chemin d'accès vers l'image à charger]
    //Charge une image depuis un fichier bmp et retourne une surface à partir de cette image.
    …
    return 0;
}
```

Pour obtenir une texture à partir d'une image, il va falloir charger une image avec la fonction précedente (qui retourne une surface) puis utiliser la fonction `SDL_CreateTextureFromSurface()` :

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_Surface* maSurface = SDL_LoadBMP("images/monImage.bmp");
    SDL_Texture* maTexture = SDL_CreateTextureFromSurface(*renderer, *maSurface)
    //[context, surface contenant l'image]
    //Charge une image dans une surface et convertit cette surface en une texture.
    …
    return 0;
}
```

Création d'une fenêtre et de son rendu SDL 2

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    SDL_Window *window;         //déclaration de la fenêtre
    SDL_Renderer *renderer;     //déclaration et association du rendu
    SDL_Surface *surface;       //déclaration de la surface
    SDL_Texture *texture;       //déclaration de la texture
    SDL_Event event;            //déclaration de l'évènement

    //initialize en testant la SDL et la SDL video
    if (SDL_Init(SDL_INIT_VIDEO) < 0)
    {
        SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "Couldn't initialize
        SDL: %s", SDL_GetError());
        return 3;
    }
    //Création de la fenêtre et du rendu avec sa taille en x et y, la
    possibilité de modifier la taille ou non, et le couple fenêtre/rendu
    de sortie
    if (SDL_CreateWindowAndRenderer(1280, 720, SDL_WINDOW_RESIZABLE,
    &window, &renderer))
    {
        SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "Couldn't create window
        and renderer: %s", SDL_GetError());
        return 3;
    }

    //gère le rendu (optionnel pour la création de fenêtre)
    while (1)
    {
        SDL_PollEvent(&event); //vérifie si un ou plusieurs évènements
        sont en cours
        if (event.type == SDL_QUIT)
        {
            break; //force la sortie de cette boucle si l'instance SDL est
            fermée
        }
        SDL_SetRenderDrawColor(renderer, 0, 0, 0, 0);
        //détermine la couleur de rendu tel que [cible, R, G, B, A]
        SDL_RenderClear(renderer); //applique au rendu la couleur définie
        SDL_RenderCopy(renderer, texture, NULL, NULL); //applique une
        texture
        SDL_RenderPresent(renderer); //produit le rendu
    }

    SDL_DestroyTexture(texture);        //détruis la texture
    SDL_DestroyRenderer(renderer);      //détruis le rendu
    SDL_DestroyWindow(window);          //détruis la fenêtre

    SDL_Quit(); //ferme toutes les instances de la SDL

    return 0;
}
```

Gérer une fenêtre avec SDL 2

Voici un liste de fonction utile dans cette optique

SDL_CreateWindow()	Crée une fenêtre
SDL_CreateWindowFrom()	Crée une fenêtre SDL à partir d'une fenêtre déjà existante
SDL_DestroyWindow()	Détruit une fenêtre
SDL_GetWindowData()	Récupère les données utilisateur associées à la fenêtre
SDL_GetWindowFlags()	Récupère les options actuelles de la fenêtre
SDL_GetWindowGrab()	Détermine si la fenêtre a le focus clavier
SDL_GetWindowPosition()	Récupère la position actuelle de la fenêtre
SDL_GetWindowSize()	Récupère la taille actuelle de la fenêtre
SDL_GetWindowTitle()	Récupère le titre actuel de la fenêtre
SDL_HideWindow()	Cache la fenêtre
SDL_MaximizeWindow()	Agrandit la fenêtre
SDL_MinimizeWindow()	Réduit la fenêtre dans la barre des tâches
SDL_RaiseWindow()	Place la fenêtre devant les autres
SDL_RestoreWindow()	Restaure la taille et la position d'une fenêtre minimisée ou maximisée
SDL_SetWindowData()	(Re)Définit les données utilisateur de la fenêtre
SDL_SetWindowFullscreen()	Passe la fenêtre en plein écran
SDL_SetWindowGrab()	(Re)Donne le focus clavier à cette fenêtre
SDL_SetWindowIcon()	(Re)Définit l'icône de la fenêtre
SDL_SetWindowPosition()	(Re)Définit la position de la fenêtre
SDL_SetWindowSize()	(Re)Définit la taille de la fenêtre
SDL_SetWindowBordered()	(Re)Définit l'affichage des bordures de la fenêtre
SDL_SetWindowTitle()	(Re)Définit le titre de la fenêtre
SDL_ShowWindow()	Affiche la fenêtre

Déclarations des points et rectangles

```
1  #include <SDL2/SDL.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int main(int argc, char *argv[])
6  {
7      …
8      SDL_Point point{ 0, 0}; //[x, y]
9      //déclare un point en position x=0 et y=0, soit en haut à gauche de
|      l'écran
10
11     SDL_Rect rectangle{0, 0, 360, 240}; //[x, y , w, h]
12     //déclare un rectangles en haut à gauche de l'écran d'une largeur de
|      360px et d'une hauteur de 240px
13
14     return 0;
15 }
```

Gérer les couleurs et modifier un fond

```
1  #include <SDL2/SDL.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int main(int argc, char *argv[])
6  {
…      …
34     SDL_SetRenderDrawColor(renderer, 0, 0, 0, 0);
35     //détermine la couleur de rendu tel que [cible, R, G, B, A]
36     SDL_RenderClear(renderer); //applique au rendu la couleur définie
37     SDL_RenderCopy(renderer, texture, NULL, NULL); //applique une texture
38     SDL_RenderPresent(renderer); //produit le rendu
…      …
47     return 0;
48 }
```

Par exemple, avec l'extrait suivant, on pourra produire un rendu rouge

```
1  #include <SDL2/SDL.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4
5  int main(int argc, char *argv[])
6  {
…      …
34     SDL_SetRenderDrawColor(renderer, 255, 0, 0, 0);
35     SDL_RenderClear(renderer);
36     SDL_RenderCopy(renderer, texture, NULL, NULL);
37     SDL_RenderPresent(renderer);
38     …
…      return 0;
47 }
48
```

Dessiner des carrés

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    SDL_Window *window;
    SDL_Renderer *renderer;
    SDL_Surface *surface;
    SDL_Texture *texture;
    SDL_Event event;

    if (SDL_Init(SDL_INIT_VIDEO) < 0)
    {
        SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "Couldn't initialize
        SDL: %s", SDL_GetError());
        return 3;
    }

    if (SDL_CreateWindowAndRenderer(1280, 720, SDL_WINDOW_RESIZABLE,
    &window, &renderer))
    {
        SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "Couldn't create window
        and renderer: %s", SDL_GetError());
        return 3;
    }

    while (1)
    {
        SDL_PollEvent(&event);
        if (event.type == SDL_QUIT)
        {
            break;
        }
        SDL_SetRenderDrawColor(renderer, 255, 0, 0, 0);
        SDL_RenderClear(renderer);
        SDL_RenderCopy(renderer, texture, NULL, NULL);
        SDL_RenderPresent(renderer);

        Draw_Rect(*surface, 100, 100, 100, 100, #000000);
        //dessine un rectangle en x=100 et y=100, avec une taille de 100px
        par 100px, et des contours noirs
        Draw_FillRect(*surface, 300, 100, 100, 100, #000000);
        //dessine un rectangle en x=300 et y=100, avec une taille de 100px
        par 100px, et de couleurs noir plein
    }

    SDL_DestroyTexture(texture);
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);

    SDL_Quit();

    return 0;
}
```

Dessiner des cercles

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char **argv)
{
    SDL_Window *window;
    SDL_Renderer *renderer;
    SDL_Surface *surface;
    SDL_Texture *texture;
    SDL_Event event;

    if (SDL_Init(SDL_INIT_VIDEO) < 0)
    {
        SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "Couldn't initialize
        SDL: %s", SDL_GetError());
        return 3;
    }

    if (SDL_CreateWindowAndRenderer(1280, 720, SDL_WINDOW_RESIZABLE,
    &window, &renderer))
    {
        SDL_LogError(SDL_LOG_CATEGORY_APPLICATION, "Couldn't create window
        and renderer: %s", SDL_GetError());
        return 3;
    }

    while (1)
    {
        SDL_PollEvent(&event);
        if (event.type == SDL_QUIT)
        {
            break;
        }
        SDL_SetRenderDrawColor(renderer, 255, 0, 0, 0);
        SDL_RenderClear(renderer);
        SDL_RenderCopy(renderer, texture, NULL, NULL);
        SDL_RenderPresent(renderer);

        Draw_Circle(*surface, 150, 150, 50, #000000);
        //dessine un cercle centré en x=150 et y=150, avec un rayon de
        50px, et des contours noirs
        Draw_FillCircle(*surface, 350, 150, 50, #000000);
        //dessine un cercle centré en x=350 et y=150, avec un rayon de
        50px, et de couleur noir plein
    }

    SDL_DestroyTexture(texture);
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);

    SDL_Quit();

    return 0;
}
```

Dessiner des points et des lignes

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    Draw_Pixel(*surface, 10, 10, #000000); //[x, y, couleur]
    //dessine un point noir en x=10 et y=10

    Draw_Line(*surface, 10, 10, 50, 50, #000000); //[x1, y2 ,x2, y2,
    couleur]
    //déclare un rectangles en haut à gauche de l'écran d'une largeur de
    360px et d'une hauteur de 240px
    …
    return 0;
}
```

SDL_Delay, faire patienter son code

```c
#include <SDL2/SDL.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    …
    SDL_Delay(60000); //[ms]
    //pause le déroulement du programme pendant un temps en ms donné,
    60000ms ici, soit 1min
    …
    return 0;
}
```