




Documentation

👤 Responsable	 Lucie
⚙️ Statut	En progression
📅 Date butoir	@23 décembre 2024
🎯 Projet	🔗 <u>Rendu</u>
🔊 Priorité	Moyenne
🏷️ Étiquette	Principal

Fonctions Principales de SDL 1.2

Initialisation et Configuration

Avant de pouvoir utiliser les fonctions SDL, il est **indispensable d'initialiser la bibliothèque**. Cela se fait généralement en utilisant `SDL_Init()`, en spécifiant les sous-systèmes à activer, comme la vidéo et l'audio. Cette fonction permet d'initialiser plusieurs sous-systèmes simultanément grâce aux flags comme `SDL_INIT_VIDEO`, `SDL_INIT_AUDIO`, ou `SDL_INIT EVERYTHING` pour tout initialiser. La source montre un exemple :

```
if(SDL_Init(SDL_INIT_EVERYTHING) != 0) {  
    fprintf(stderr, "Error in SDL_Init : %s\n", SDL_GetError());  
    exit(EXIT_FAILURE);  
}
```

Une fois que vous avez terminé d'utiliser la SDL, il est important de la fermer correctement avec `SDL_Quit()` pour libérer les ressources allouées et éviter les fuites mémoire.

Gestion de la Fenêtre

La création d'une fenêtre d'application se fait avec `SDL_SetVideoMode()`. Cette fonction est fondamentale car elle initialise l'affichage graphique du jeu.

Exemple :

```
SDL_Surface * ecran = NULL;
if((ecran = SDL_SetVideoMode(800, 600, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF | SDL_FULLSCREEN)) == NULL) {
    fprintf(stderr, "Error in SDL_SetVideoMode : %s\n", SDL
_GetError());
    SDL_Quit();
    exit(EXIT_FAILURE);
}
```

`SDL_WM_SetCaption()` permet de définir le titre de la fenêtre.

Surfaces et Affichage

SDL 1.2 utilise le concept de **surfaces** (`SDL_Surface`) pour gérer les zones de pixels.

- `SDL_FillRect()` Remplit un rectangle avec une couleur.
- `SDL_BlitSurface()` Copie une surface sur une autre.
- `SDL_Flip()` met à jour l'écran en affichant le contenu de la surface de dessin.
- `SDL_FreeSurface()` libère la mémoire occupée par une surface.

Gestion des Événements

SDL utilise une structure `SDL_Event` pour représenter les différents types d'événements (mouvements souris, fenêtre etc...).

- `SDL_PollEvent()` Récupère les événements en attente dans la file d'événements SDL.
- Accessible via le champ `event.type`, qui permet d'identifier la nature de l'événement (`SDL_KEYDOWN`, `SDL_MOUSEMOTION`, etc.).
- Des champs supplémentaires dans `SDL_Event` existent comme les coordonnées de la souris ou la touche du clavier enfoncée.

SDL_ttf (Gestion du Texte)

- `TTF_Init()` doit être appelé avant toute utilisation des fonctions.
- `TTF_OpenFont()` Charge un fichier de police TTF avec une taille spécifique.

- `TTF_RenderText_Solid()` pour le rendu.
- `TTF_CloseFont()` libère la mémoire utilisée par une police.
- `TTF_Quit()` Appelé à la fin pour nettoyer les ressources.

SDL_mixer

Supporte plusieurs formats audio comme WAV, MP3, OGG.

- `Mix_LoadMUS()` charge un fichier musical en mémoire.
- `Mix_PlayMusic()` avec options de répétition.
- `Mix_HaltMusic()` arrête immédiatement la lecture.
- `Mix_VolumeMusic()` de 0 (muet) à 128 (maximum).
- `Mix_FreeMusic()` libère la mémoire.
- `Mix_CloseAudio()` ferme le système audio.
- `Mix_GetError()` pour le débogage.

SDL_image

- `SDL_SoftStretch` permet de redimensionner l'image .
- `SDL_surface` Pour la gestion de l'affichage, des images, de l'écran de la fenêtre.

Nettoyage (SDL 1.2, SDL_mixer 1.2, SDL_ttf)

Libération des ressources.

```
SDL_FreeSurface(ecran);
Mix_CloseAudio();
Mix_Quit();
TTF_CloseFont(police);
TTF_Quit();
SDL_Quit();
```

Compilation

```
$ gcc -o reigns src/*.c -I"C:/msys64/mingw64/include" -I"./in
```

