Einführung in SDL unter Linux

bonynix

8. März 2009

Inhaltsverzeichnis

1 Was ist SDL

SDL ist eine Bibliothek, welche die GUI-Programmierung ehrheblich erleichtert. Desweiteren ist SDL sehr portabel wodurch sie eine beliebte Bibliothek für Spieleprogrammierer geworden ist. Dieses Tutorial wird den praktischen Umgang mit SDL zeigen.

1.1 Was braucht man für die SDL-Programmierung

Für die SDL-Programmierung wird ein Compiler, ein Texteditor und die Entwicklungsbibliotheken von SDL benötigt.

Installieren sie *die SDL-Libaries* [1] unter den .deb Packetsystem einfach mit

```
sudo aptitude install libsdl1.2-dev
```

Der Wahl des Texteditors bleibt völlig ihnen überlassen genauso wie die des Compilers. Ich nehme aber an, dass sie den gcc genommen haben, deswegen folgt ein kleiner Einstieg zur SDL-Programmcompilierung mit dem Gcc.

1.1.1 Compilieren mit dem Gcc

Compiliert wird mit dem Befehl:

```
gcc ‘sdl-config --libs --cflags‘ -o program program.c
```

oder für C++:

```
g++ -o program program.cpp -lSDL
```

[1] Es gibt noch weitere SDL-Bibliotheken, welche man später noch installieren muss, darauf werde ich aber noch eingehen.

1.2 Erstes Program

Wir wollen uns SDL anhand eines ersten Programs näher betrachten.

```
#include <SDL/SDL.h>

int main(void)
{
      SDL_Surface *screen;

      if( SDL_Init(SDL_INIT_VIDEO) != 0) {
            printf("Error: %s\n",SDL_GetError());
            return EXIT_FAILURE;
      }

      atexit(SDL_Quit);

      screen = SDL_SetVideoMode(640, 480, 0, SDL_SWSURFACE);

      if(screen == NULL) {
            fprintf(stderr,"Error: %s\n",SDL_GetError());
            return EXIT_FAILURE;
      }

      SDL_Delay(1000);

      return 0;
}
```

1.2.1 Einbinden der Bibliotheken

Das Programm wird mit einem

`#include <SDL/SDL.h>`

begonnen. *Dies bindet die SDL-Libaries in das Programm ein.* [2] Unter Windowsumgebungen hat man oft die SDL.h-Bibliothek in einem extra SDL-Ordner deswegen die erweiterte Pfadangabe, unter Linux könnte man das SSDL/ëinfach weglassen.

1.2.2 Deklaration und Initialisierung von Variablen

Kommen wir nun zur Deklaration von SDL-Strukturen, welche sich eigentlich wie alle andere Deklarationen handhaben lässt.

[2] Es ist natürlich auch mit #includeSSDL.h"möglich, wenn sich die Bibliothek im gleichen Ordner befindet.

```
SDL Surface *screen;
```

Diese Zeile deklariert schlicht die Struktur ßcreen", inder man später Bilder speichern kann. Weitere Strukturen in SDL sind zum Beispiel SDL_Rect und SDL_Event, auf die ich im weiteren Verlauf des Artikles noch zu sprechen komme.

Initialisiert werden die eigen SDL_Datentypen auch normal, wobei es gleich zum Anfang zu einer Ausnahme kommt.
Diese Ausnahme ist das SDL_Surface ßcreen".

```
screen = SDL_SetVideoMode(640, 480, 0, SDL_SWSURFACE);
```

Hier wird die Hauptoberfläche des Programs initialisiert, welche es nur einmal geben sollte. Gegliedert ist die Zuweisung folgend:

```
screen = SDL_SetVideoMode( Fensterbreite(x), Fensterhöhe(y),
                           Farbtiefe, Spezielle Parameter | ... );
```

Als erstes kann man Fensterbreite sowie Höhe übergeben, danach folgt die *Farbtiefe*[3]. Wenn man 0 eingibt versucht SDL den bestmöglichstens Wert zu übergeben. Schließlich kommen wir zu den speziellen Parametern. Man kann hier beliebig viele Flags übergeben welche zum Beispiel folgende wären:

Paramter	**Wirkung**
SDL_FULLSCREEN	Fenster breitet sich über den ganzen Bildschirm aus(Vollbild)
SDL_SWSURFACE	Sagt SDL für Surfaces den Arbeitsspeicher zu nutzen
SDL_HWSURFACE	Sagt SDL für Surfaces den Grafikkartenspeicher zu nutzen
SDL_RESIZABLE	Kann ein in der Größe ändernes Fenster erzeugen

Diese Flags sollten fürs Erste genügen, allerdings sei gesagt das es noch mehrere gibt.

[3] gemessen in Bit

1.2.3 Starten und Beenden von SDL

Kommen wir nun zum Starten von SDL.

```
if( SDL_Init(SDL_INIT_VIDEO) != 0) {
    printf("Error: %s\n",SDL_GetError());
    return EXIT_FAILURE;
}
```

Diese paar Zeilen Code laden die einzelnen *SDL-Module* [4] in das Program hinein und überprüfen den Vorgang gleich nach Fehlern. Neben den SDL_Video-Modul gibt es noch verschieden weiter, wie z.B:

Modul	Wirkung
SDL_INIT_JOYSTICK	Initialisiert das Joystick-Modul
SDL_INIT_CDROM	Initialisiert das Cdrom-Modul
SDL_INIT_AUDIO	Initialisiert das Audio-Modul
SDL_INIT_TIMER	Initialisiert das Zeit-Modul
SDL_INIT_EVERYTHING	Initialisiert alles

Wiederum sei angemerkt, dass ich 1-2 Module bewusst weggelassen habe. ;)

Beendet wird SDL einfach mit

```
atexit{SDL_Quit);
```

wobei man auch nur SDL_Quit am Ende seines Programmes schreiben könnte, diese Notation aber deutlich unsicherer ist, da man relativ leicht vergisst SDL zu beenden. Atexit nimmt verschiedene Funktionen auf und führt sie nach Beendigung des Programms in umgekehrter Reihenfolge aus. (diese Funktion gehört übrigens zu C)

1.2.4 SDL_Delay

Die Funktion SDL_Delay lässt das Programm für x millisekunden pausieren. Mit dieser Funktion kann man also relativ leicht die Prozessorlast verringern und auch programmgewollte Verzögerungen erzeugen. Hier nochmal die Funktion im Überblick:

```
SDL_Delay(Uint32 ms);
```

[4] Man spricht auch von Untersystemen

```
wie z.B

SDL_Delay(1000); // Wartet eine Sekunde
```

Uint32 bedeutet, dass ein Integer ohne *Vorzeichen* [5] mit *32 Bit-Länge* [6] übergeben wird. Die 32 Bit sind für die Plattformunabhängigkeit von höherer Bedeutung uns haben diese ganze Zahlen nicht *zu interessieren* [7] weswegen ich sie in den nächsten Funktionsdeklaration wegfallen lassen werde.

1.3 Fenstereigenschaften

Die SDl-Bibliotheken können auch die Fenstereigenschaften eines jenen Programmes bestimmen. Mit Fenstereigenschaften meine ich den Namen, die Beschreibung und das Icon, welches man von seinem Fenstermanager angezeigt bekommt. Fangen wir mit der Beschreibung und des Titels eines Fensters an.

```
void SDL_WM_SetCaption(const char Beschreibung, const char Titel);

wie z.B

SDL_WM_SetCaption("Spiel - Ein tolles Spiel", "Spiel");
```

Das Icon lädt man folgend:

```
void SDL_WM_SetIcon(SDL_Surface *ICON, Uint8 *position);

wie z.B

SDL_WM_SetIcon(IMG_Load("/home/hans/testbild.bmp"), 0);
```

ACHTUNG:
SDL_WM_SetIcon muss vor SDL_SetVideoMode aufgerufen werden

[5] kein Minus
[6] 4 Byte
[7] Aus der Sicht eines Startes mögen sie mehr verwirren als zu helfen.

1.4 Übung

Um das Wissen zu festigen werden am Ende jedes Kapitels aufgaben gestellt die es mit dem erlangten Wissen zu lösen gilt. Lösungen sind am Ende aufgelistet oder auch direkt im data-Ordner ansehbar.

1. Schreiben sie ein Program welches SDL mit Vollbildmodus startet und 3 Sekunden wartet.
2. Was ist SDL_Surface* und wozu verwendet man es hauptsächlich ?
3. Mit welchen Befehl initialisiert man SDL ?

2 Laden und Positionierung von Bildern

Nachdem sie nun den ersten, relativ langweiligen Teil absolviert haben, kommen wir in diesem Kapitel zu sichtbaren Erfolgen, welche damit gleich interessanter werden.

2.1 Laden von Bildern

Hierbei können sie schon gleich auf die Erfahrungen aus dem ersten Kapitel zurückgreifen, da sie dort ja schon ein Surface, in den die Bilder gespeichert werden, deklariert haben (screen). Deklariert wird also folgend:

```
SDL_Surface *img;
```

Nun wollen wir aber endlich auch mal dieses Bild initialisieren.

```
img = SDL_LoadBMP("/home/peter/pfad/zum/bild");
```

Es soll gesagt sein das SDL inmoment in ihrer Standardbibliothek nur bmp unterstützt. Deswegen greifen viele zur SDL_image-Bibliothek da sie weitere Bildformate unterstütz, wie zum Beispiel GIF, JPEG, LBM, PCX, PNG, PNM, TGA, TIFF, XCF, XPM. Diese wird mit

```
sudo aptitude install libsdl-image-dev
```

installiert. Und folgend eingebunden

```
#include <SDL/SDL_image.h>
```

Natürlich muss man dem Compilier die neue Bibliothek noch hinzulinken, wobei man einfach ein

```
-lSDL_image
```

zu den Parametern schreibt. Nachdem wir uns jetzt optimal auf die neue Bibliothek angepasst haben, wollen wir sie auch benutzten. Zu den neuen benutzbaren Bildern zählen .png's, .jpgs's, .gif's, .pnm's und .xpm's um die bekanntesten mal zu nennen. Ein Bild wird mit dieser Funktion geladen:

```
img = IMG_Load("Pfad zum Bild");
```

Natürlich sollte man diese Zuweisung auf Fehler überprüfen:

```
if(img == NULL) {
        fprintf(stderr, "Konnte Bild nicht laden\n");
        return 1;
}
```

2.2 Anzeigen von Bildern

Nachdem wir nun das Bild erfolgreich initialisiert haben, wollen wir es uns natürlich auch anzeigen lassen. Dies wird üblicherweise mit

```
SDL_BlitSurface( Bild, Bildposition, Ziel(Bildschirm), Zielposition)
```

Dieses würde dann in unserem Beispiel so

```
SDL_BlitSurface( img, 0, screen, 0 );
```

aussehen. Natürlich könnte man diese Funktion auch wieder auf Fehler überprüfen was aber unüblich ist.

2.3 Positionieren von Bildern

Mit der geraden eben besprochenen Blit-Funktionen ist es bisher allerdings nur möglich die Bilder im oberen rechten Eck des Bildschirms anzeigen zu lassen. Um dies zu ändern bedient man sich sogenannten SDL_Rect's. Ein SDL_Rect ist wieder eine Struktur die SDL bereitstellt und enthält folgende Positionsangaben: x, y, w(weite), h(höhe).

Deklariert und initialisiert wird sie folgend:

```
SDL_Rect img_position;

img_position.x = 10;
img_position.y = 10;
img_position.w = 12;
img_position.h = 13;
```

Nun können wir die Positionsangabe einfach SDL_BlitSurface übergeben:

```
SDL_BlitSurface( img, 0, screen, img_position );
```

2.4 Bilderneuerung

Nachdem wir nun unser Bild auf das Hauptfenster kopiert haben, wollen wir es natürlich auch sehen, wozu das Hauptfenster geupdatet werden muss. Um dies zu erreichen gibt die folgenden 3 Standartmethoden:

- SDL_UpdateRect(Bildschirm, x, y, w, h);
- SDL_UpdateRects(Bildschirm, anzahl der Rects, *rect);
- SDL_Flip(Bildschirm);

2.4.1 SDL_Flip()

Die Funktion sieht folgendermaßen aus:

int SDL_Flip(SDL_Surface* screen);

2.5 Übungen

Wie gewohnt gibt es wieder ein paar Übungen

1. Schreiben sie ein Program welches das Bild "Test.pngäuf den Bildschirm zentral anzeigt.
2. Schreiben sie ein Program welches das Bild "Test.png"von links nach rechts bewegt bis es das Ende des Bildschirm erreicht hat.
3. Schreiben sie das Programm aus der letzten Übung so um, dass das Bild beim verschieben keine Spuren hinterlässt.
4. Wozu benötitgt man die SDL_image-Bibliothek ?

3 Benutzereingaben verarbeiten

Auch die Benutzereingabenverwaltung unter SDL ist relativ simpel. Zurallerst deklariert man mit:

```
SDL\_Event event;
```

Ein sogenanntes 'Event' in welches man dann später auslesen kann.

3.1 Tastatur

3.2 Maus

Eine Simple Abfrage ob die Mausgedrück wurde lässt sich mit folgenden Code-Stück realisieren.

```
while(SDL_PollEvent(&event)) {
if(event.type == SDL_MOUSEBUTTONDOWN) {
if(event.button.button == SDL_BUTTON_RIGHT) {
```

4 Weitere Bibliotheken von SDL

4.1 SDL_Net

Die Netzwerkbibliothek von SDL.

4.2 SDL_TTF

Die TTF-Bibliothek von SDL mit der TTF-Grafiken angezeigt werden können.

4.3 SDL_Mixer

Die Audiobibliothek von SDL, welche den Sound auf mehreren Kanälen abspielen kann, sowie noch weitere *Audiocodecs* [8] unterstützt.
Folgendes Beispielprogramm spielt eine WAV-Datei ab während SDL eine Sekunde wartet:

```
#include <SDL/SDL.h>
#include <SDL/SDL_mixer.h>

Mix_Chunk *sound;
SDL_Surface *screen;

void init_soundeffect(void)
{
    if(sound == NULL) {
        sound = Mix_LoadWAV("sound.wav");
        if(sound == NULL) {
            printf("Unable to load WAV file: %s\n", Mix_GetError());
        }
    }
}

void soundeffect(void)
{
    int channel;

    /* Play sound file, and capture the channel */
    channel = Mix_PlayChannel(-1, sound, 0);
    if(channel == -1) {
        printf("Error: Unable to play soundeffect: %s\n", Mix_GetError());
    }
}

int main(int argc, char **argv)
{
    int fullscreen = 1;
```

[8] wie zum Beispiel .mp3

```
    /* Init SDL */
    if( SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO ) != 0) {
        printf("Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    /* setting up sound */
    if(Mix_OpenAudio(22050, AUDIO_S16SYS, 2, 4096) != 0) {
        printf("Unable to initialize audio: %s\n", Mix_GetError());
        exit(1);
    }

    atexit(SDL_Quit);
    atexit(SDL_CloseAudio);
    atexit(Mix_CloseAudio);


    /* setting up screen */
    screen = SDL_SetVideoMode(80, 80, 0, SDL_SWSURFACE | SDL_DOUBLEBUF |
                              SDL_INIT_AUDIO | (fullscreen ? SDL_FULLSCREEN : 0));
    if(screen == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }


    init_soundeffect();
    soundeffect();

    SDL_Delay(1000);

    Mix_FreeChunk(sound);
    return EXIT_SUCCESS;
}
```

5 Schlusswort

Als letztes sollte man noch erwähnen das es für SDL in anderen Sprachen sehr schöne Bindings gibt. Sprachen mit Bindings wären zum Beispiel:

- Python
- Perl
- Haskell
- Ruby
- Vala
- D
- C#
- ...

Nun wünsche ich ihnen aber viel Spaß mit den neu erlangten Wissen ;).

6 Quellen

6.1 Quellen

- http://www.libsdl.org/projects/SDL_image/
- http://www.libsdl.org/projects/SDL_mixer/
- http://de.wikipedia.org/wiki/Simple_DirectMedia_Layer
- http://de.wikibooks.org/wiki/SDL

A Lösungen

A.1 Was ist SDL

1. Ein Beispiel wie die Übung 1 aussehen kann:

```
#include <SDL.h>

int main(void)
{
    SDL_Surface *screen;

    if( SDL_Init(SDL_INIT_VIDEO) != 0) {
        printf("Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    atexit(SDL_Quit);

    screen = SDL_SetVideoMode(640, 480, 0, SDL_FULLSCREEN);

    if(screen == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    SDL_Delay(3000);

    return 0;
}
```

2. Ein SDL_Surface* ist ein Zeiger auf eine Struktur welche SDL_Surface heißt. In dieser speichert SDL verschiedene Werte zum Bild wie beispielsweise Breite, Höhe, ...

3. SDL_Init();

A.2 Laden und Positionieren von Bildern

1. Ein Beispiel wie die Übung 1 aussehen kann:

```
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>

int main(void)
{
    SDL_Surface *screen;
    SDL_Rect central;
    SDL_Surface *img_test;

    if( SDL_Init(SDL_INIT_VIDEO) != 0) {
        printf("Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    atexit(SDL_Quit);

    screen = SDL_SetVideoMode(640, 480, 0, 0);

    if(screen == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    /* loading test.png */

    img_test = IMG_Load("test.png");
    if(img_test == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    /* locating the img in the middle of the screen */

    central.x = screen->w/2-img_test->w/2;
    central.y = screen->h/2-img_test->h/2;

    SDL_BlitSurface(img_test, 0, screen, &central);
    SDL_Flip(screen);

    SDL_Delay(2000);

    return 0;
}
```

2. Ein Beispiel wie die Übung 2 aussehen kann:

```
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>

int main(void)
{
    SDL_Surface *screen;
    SDL_Rect right;
    SDL_Surface *img_test;
    int dx, end_screen;

    if( SDL_Init(SDL_INIT_VIDEO) != 0) {
        printf("Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    atexit(SDL_Quit);

    screen = SDL_SetVideoMode(640, 480, 0, 0);

    if(screen == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    /* loading test.png */

    img_test = IMG_Load("test.png");
    if(img_test == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    /* locating the img in the right of the screen */

    right.x = img_test->w/2;
    printf("right.x = %d",right.x);
    right.y = screen->h/2-img_test->h/2;

    SDL_BlitSurface(img_test, 0, screen, &right);
    SDL_Flip(screen);

    /* initialize how fast the img should move */
    dx = 10;

    /* moving the img */

    while(right.x < 450) {
```

```
        right.x += dx;
        SDL_BlitSurface(img_test, 0, screen, &right);
        SDL_Flip(screen);
        SDL_Delay(500);
    }

    return 0;
}
```

3. So könnte man Übung 3 schreiben:

```
#include <SDL/SDL.h>
#include <SDL/SDL_image.h>

int main(void)
{
    SDL_Surface *screen;
    SDL_Surface *background;

    background = IMG_Load("background.png");
    if(background == NULL) {
        fprintf(stderr,"Error: %s\n",SDL_GetError());
        return EXIT_FAILURE;
    }

    [...]

    /* moving the img */

    while(right.x < 450) {
    /* overblitting the old rect_position */
        SDL_BlitSurface(background, &right, screen, &right);
        right.x += dx;
        SDL_BlitSurface(img_test, 0, screen, &right);
        SDL_Flip(screen);
        SDL_Delay(500);
    }

    return 0;
}
```

4. Die SDL_image-Bibliothek wird benötitgt um Grafiken in anderen Formaten als Bmp in SDL einzulesen wie z.B .png oder .jpg

A.3 Benutzereingaben verarbeiten

1. SDL_PollEvent(SDL_Event *event) bewirkt dass das übergebene Event mit einem Ereignisevent initialisiert wird.

A.4 Weitere Bibliotheken von SDL

1. kommt demnächst [...]