SDL (Version 2.x) mit Windows und CodeBlocks

Klaus Kusche, 2021

Zutaten

Du brauchst 2 Dinge:

- Das offizielle **SDL2-Installationspaket**, Windows-Version für *Entwicker*
  (***"Development Libraries"***, *nicht "Runtime Binaries" oder "Source Code"!*)
  mit *MinGW* (*nicht* die Visual-Studio-Variante!).
  Derzeit ist das https://www.libsdl.org/release/SDL2-devel-2.0.14-mingw.tar.gz
  auf https://www.libsdl.org/download-2.0.php
  (eine höhere 2.0.x-Version, also z.B. 2.0.15, sollte genauso funktionieren).

  In den Rechnerräumen der DHGE sollte irgendeine Version bereits fertig ausgepackt
  in **`C:\SDL2-2.0.x\`** liegen (sonst bitte runterladen und am besten auf **`D:\`**
  entpacken), auf privaten Rechnern auch einfach irgendwohin *entpacken*.

  Das Entpacken eines …**`.tar.gz`**-Archivs lässt sich grafisch in Windows
  nur mit zusätzlich installierten Archiv-Programmen (WinRAR, 7zip) bewerkstelligen,
  aber im **`cmd.exe`** geht es bei aktuellem Windows 10 auch mit Windows-Bordmitteln:
  `cd` in das Download-Verzeichnis, dann dort **`tar xf SDL2-….tar.gz`** eingeben.

  Achtung: Der *SDL-Zip-File von meiner Webseite für DevCpp an der NTA* enthält zwar
  auch alles, was nötig ist, aber in *ganz anderer Anordnung und unter anderen Namen*
  als der offizielle Download. Man bekommt zwar auch damit ein SDL-Grafikprogramm
  problemlos unter CodeBlocks zum Laufen, wenn man weiß, was man tun muss,
  aber diese Anleitung passt dafür überhaupt nicht!
- Die beiden von mir programmierten Files **`sdlinterf.c`** und **`sdlinterf.h`**,
  die eine **Zwischenschicht** zwischen deinem Programm und der SDL bilden,
  damit du nicht direkt auf die SDL-Funktionen zugreifen musst.

Anlegen und Compilieren von SDL-Programmen

- Leg in CodeBlocks ein ***neues Projekt*** an,
  und zwar als *Konsol-Projekt* ("*Console application*"):
  Wir wollen ja Ausgaben und Fehlermeldungen zur Laufzeit im DOS-Fenster sehen!

  Der *Projektname* (wird auch Name des **`.exe`**-Files!) und alle im Projekt verwendeten
  Dateinamen sollen *keine Umlaute oder Zwischenräume* enthalten.

  Stelle als *Projekt-Sprache C* ein, wenn das Projekt *keinen einzigen* C++-File enthält,
  und *stelle C++ ein*, wenn du ein *C++-Programm* schreiben willst!

  CodeBlocks legt als *Speicherort* für das Projekt einen neuen, leeren Ordner
  unter dem beim Anlegen des Projektes angegebenen Ordner an.
- ***Kopiere*** meine beiden **`sdlinterf`**-Dateien *in diesen neuen Projekt-Ordner*
  (und ev. auch die Musterlösung oder das Demo-Programm,
  das du zum Ausprobieren verwenden willst, also z.B. **`demo0.c`** oder **`gra.cpp`**).

- Du kannst entweder im Editor den Code der Demo oder Musterlösung in das automatisch erzeugte **`main.c`** *hineinkopieren* oder das automatisch erzeugte **`main.c`** aus dem Projekt *entfernen* (links in der Projekt-Ansicht: Rechtsklick auf **`main.c`**, *"Remove file from Project"*) und stattdessen den Demo-**`.c`**-File *ins Projekt aufnehmen* (links in der Projekt-Ansicht: Rechtsklick auf das Projekt selbst, *"Add files..."*).

  Achtung: Wenn dein Projekt einen **`main.c`** (nicht **`main.cpp`**) enthält und du ein **`.cpp`**-Grafikprogramm hast, musst du den **`.cpp`**-File *statt* **`main.c`** ins Projekt geben, denn C++-Code in ein **`main.c`** kopiert lässt sich natürlich nicht compilieren!

- Füge auch **`sdlinterf.c`** und **`.h`** (und ev. weitere *eigene* **`.c`** / **`.h`**-Files) zum Projekt *dazu*, indem du links in der Projekt-Ansicht einen Rechtsklick auf das Projekt selbst machst und *"Add files..."* wählst.

- Dein C/C++-Programm muss ein **`#include`** auf meinen **`sdlinterf.h`** machen (mit **`"sdlinterf.h"`**, *nicht* mit **`<sdlinterf.h>`** !).

- Geh in den ***Compiler-Einstellungen*** des Projektes (links in der Projekt-Ansicht: Rechtsklick auf das Projekt selbst, *"Build options..."*)...

  - ... zuerst auf das dritte Karteiblatt *"Search directories"*. Trage dort bei *"Compiler"* mit [Add] das *Include-Verzeichnis* der SDL-Installation ein:

    *SDL2-Installationspfad***`\i686-w64-mingw32\include\`** .

  - Trage genauso bei *"Search directories"* / *"Linker"* das *Lib-Verzeichnis* der SDL-Installation ein:

    *SDL2-Installationspfad***`\i686-w64-mingw32\lib\`** .

  Hinweis 1:

  Beide Pfade sind für den **32-Bit**-MinGW-Compiler (CodeBlocks bis incl. **Version 17.12**). Solltest du den **64-Bit**-MinGW-Compiler verwenden (CodeBlocks ab **Version 20.03**), wird **`\i686-w64-mingw32\`** durch **`\x86_64-w64-mingw32\`** ersetzt!

  Hinweis 2:

  Wenn du das *SDL-Installationspaket* irgendwo *zentral* auf **`C:`** oder **`D:`** ausgepackt hast, sag *"nein"*, wenn er fragt, ob er die Pfade relativ speichern soll.
  Wenn du die SDL-Installation direkt in deinen Projektordner entpackt hast, oder wenn du sie gemeinsam mit deinem Projekt auf demselben *Stick oder Netzlaufwerk* liegen hast, sag *"ja"*.

- Geh dann in den *"Build options..."* auf das vorige Karteiblatt "*Linker Settings*". Trage dort im rechten Feld *"Other linker options:"* folgenden Text ein:

  `-static -lmingw32 -lSDL2main -lSDL2`
  `-Wl,--no-undefined -lm -ldinput8 -ldxguid -ldxerr8 -luser32`
  `-lgdi32 -lwinmm -limm32 -lole32 -loleaut32 -lshell32`
  `-lversion -luuid -static-libgcc -lhid -lsetupapi`

  Bei **C++-Programmen** kommt noch **`-static-libstdc++`** dazu.

  (im Unterschied zur offiziellen SDL-Doku *ohne* **`-mwindows`**,
  wir schreiben ja keine reine GUI-Anwendung, sondern wollen auch
  den **`stdout`**-Output unseres Programms im DOS-Fenster sehen!)

Mit diesen Einstellungen sollte sich eine Musterlösung bzw. dein Programm fehlerfrei compilieren, linken und starten lassen.

Verwendung meiner **sdlinterf.h** und **sdlinterf.c**

Welche Funktionen wie aufgerufen werden müssen, um etwas graphisch anzuzeigen, entnimmst du den *Kommentaren* zu den von mir bereitgestellten Funktionen in **sdlinterf.h** und meinen *Beispiel-Programmen* **demo0.c**, **demo1.c**. und **demo2.c**.

Du kannst die Fenstergröße **SDL_X_SIZE** und **SDL_Y_SIZE** in **sdlinterf.h** an deinen Computer *anpassen*, wenn das Fenster für deine Display-Auflösung zu klein / zu groß ist. Danach musst du *alles neu compilieren*, auch **sdlinterf.c** !

Auf manchen Rechnern kommt es trotz korrekter Programme zu *seltsamen Anzeige-Fehlern* (beispielsweise gehen Teile der Anzeige verloren, oder schon gelöschte Pixel bleiben übrig), vor allem bei *schnell bewegten Grafiken* (das Problem wurde bisher vor allem auf *Apple Mac* beobachtet). In diesem Fall kann es helfen, in **sdlinterf.c** in der Funktion **sdlInit** den alternativen Aufruf von **SDL_CreateRenderer** (mit **SDL_RENDERER_SOFTWARE**) *statt* dem standardmäßigen Aufruf zu verwenden.

SDL2-Projekte & mein **sdlinterf** mit CMake bzw. CLion

Hier haben zwei Studenten dokumentiert, wie man die SDL und mein **sdlinterf** in ein *CMake-Projekt* (unter *Linux* oder unter *Windows*) einbindet:

https://github.com/ZeroPointMax/sdlDoc

SDL2-Programme & mein **sdlinterf** "handcompilieren" unter Linux & Mac

1.) Die *SDL2-Pakete* in der *Entwickler-Version* (oft "**devel**" im Namen) installieren (wie die Pakete genau heißen und wie man sie installiert ist systemabhängig).

2.) Unter Unix/Linux und auch unter MacOS sollte eine SDL-Installation auch den Befehl **sdl2-config** installieren. Er liefert die *zum Compilieren nötigen Informationen* (Verzeichnisse, Libraries, Compiler-Optionen, ...). Genauer:

- **sdl2-config --cflags** liefert die Optionen für den C- oder C++-Compiler (z.B. das Include-Verzeichnis mit den SDL-Headern)
- **sdl2-config --libs** liefert die Optionen für den Linker (die Namen der SDL-Libraries und deren Verzeichnis)

Prüfe, ob **sdl2-config** funktioniert!

3.) Der **sdl2-config**-Befehl lässt sich mittels ` (= "verkehrtes" Anführungszeichen) direkt in den *Compiler-Aufruf auf der Befehlszeile* einbauen.

Für C (alles in *einer* Zeile):

gcc `sdl2-config --cflags` *andere Compiler-Optionen* **-o** *exe-File-Name alle C-File-Namen nacheinander* **sdlinterf.c `sdl2-config --libs`**

Für *C++-Programme* **g++** statt **gcc** als Compiler-Befehl verwenden.

Wenn man **-o** *exe-File-Name* weglässt, heißt der erzeugte exe-File **a.out** .

Statt **`sdl2-config --cflags`** ist auch **$(sdl2-config --cflags)** erlaubt (`...` ist Steinzeit-Unix, **$(...)** ist die moderne Schreibweise für denselben Zweck).