SDL (Version 2.x, nicht 3.x !!!) mit Windows und CodeBlocks

Klaus Kusche, 2026

Zutaten

Du brauchst 2 Dinge:

- Das offizielle **SDL2-Installationspaket**, Windows-Version für *Entwicker*
  (***"Development Libraries"***, *nicht* *"Runtime Binaries" oder "Source Code"!*)
  für *MinGW* (*nicht* die Visual-Studio-Variante!).
  Derzeit ist das https://github.com/libsdl-org/SDL/releases/download/release-2.32.6/SDL2-devel-2.32.6-mingw.zip auf
  https://github.com/libsdl-org/SDL/releases/tag/release-2.32.6
  (eine höhere oder ältere 2.x-Version sollte genauso funktionieren).
  Siehe auch https://www.libsdl.org/ .

  Achtung: SDL ***Version 2.x***, ***nicht*** Version ***3.x*** ! Mein **`sdlinterf`** kann ***kein*** SDL 3.x !

  In den Rechnerräumen der DHGE sollte irgendeine Version bereits fertig ausgepackt auf den Rechnern liegen:
  - Auf ganz *neu installierten Rechnern* in **`C:\Library\SDL2-devel-2.30.8-mingw`**
  - Auf Systemen mit älterer Installation eventuell direkt unter **`C:\`** oder **`D:\`**

  Sonst einfach irgendwohin *entpacken* (ebenso auf privaten Rechnern).
  Zur Minimierung des Klick-Aufwandes beim Einrichten in Codeblocks empfehle ich
  - *Entweder möglichst weit oben* in der Ordner-Hierarchie (direkt unter **`C:`**, **`D:`** oder **`Z:`**)
    (wenn du die SDL für *mehrere* Codeblocks-Projekte nutzen willst)
  - Oder *direkt im Projekt-Ordner* deines CodeBlocks-Projektes
    (wenn du die SDL *nur für ein Projekt* nutzen willst)
- 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 Datei- und Ordner-Namen 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 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`**).

(Fortsetzung nächste Seite ...)

- Du kannst entweder im Editor den Code der Demo oder der 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..."*).
- Füge auch **`sdlinterf.c`** und **`.h`** (und eventuell 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"`** (in Anführungszeichen!) *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`\x86_64-w64-mingw32\include` .
  - Trage genauso bei *"Search directories"* / *"Linker"* das *Lib-Verzeichnis* der SDL-Installation ein:

    SDL2-Installationspfad`\x86_64-w64-mingw32\lib` .

  Hinweis 1:

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

  Für **CodeBlocks 25.03** ist *derselbe SDL-Download* und *dieselbe Installations-Anleitung* zu verwenden wie für **CodeBlocks 20.03**.

  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 (*zweite*) 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.

(Fortsetzung nächste Seite ...)

SDL2-Projekte & mein `sdlinterf` mit CMake bzw. CLion usw.

Hier haben zwei DHGE-Studenten im Jahr 2020 dokumentiert, wie man die SDL und mein **`sdlinterf`** in ein *CMake-Projekt* (unter *Linux* oder unter *Windows*) einbindet:

https://github.com/ZeroPointMax/sdlDoc
(irgendwo gibt's das sicher auch aktueller, aber wo?)

Meines Wissens gibt es in den jüngeren DHGE-Jahrgängen auch fertige **`.bat`**-*File-Vorlagen* für das Kompilieren auf der **`cmd`**-Befehlszeile unter Windows (aber wo?).
Deren Anpassung geht schneller als das Einrichten der SDL unter Codeblocks, und ein Batchfile ist unabhängig von der Entwicklungsumgebung verwendbar (d.h. sollte auch mit Visual Studio Code usw. funktionieren).

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

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

MacOS: Der *Selbstbau* der SDL wie auf der SDL-Homepage beschrieben ist *umständlich und normalerweise nicht zielführend*.

Es ist einfacher, sich den Download-Manager eines der bekannten Mac-Open-Source-Software-Repositories zu installieren und damit ein ***fertig gebautes** SDL-Paket* herunterzuladen und zu installieren. Die Studenten verwenden meist Homebrew: https://formulae.brew.sh/formula/sdl2

Tipp: **`brew`** verändert **`PATH`**. Damit der geänderte **`PATH`** *wirksam* wird, muss man das *Terminal-Fenster einmal schließen und wieder öffnen*.
Also: *brew installieren - zu/auf - brew install sdl2 - zu/auf - eigenes Programm bauen*

(MacPorts, Fink oder Anaconda bieten auch fertige SDL-Pakete an)

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 **`$(...)`** direkt in den *Compiler-Aufruf auf der Befehlszeile* einbauen.

Für C (alles in *einer* Zeile):

`gcc` *andere Compiler-Optionen* **`$(sdl2-config --cflags) -o`** *exe-File-Name alle .c- und .cpp-Files 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`** .

Auf Apple Mac:
Den Hinweis betreffend **`SDL_RENDERER_SOFTWARE`** im nächsten Kapitel beachten!!!

(Fortsetzung nächste Seite ...)

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

a) Aufruf:

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**.

b) Fenstergröße & Fullscreen:

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* (Button "Rebuild all"), auch **sdlinterf.c** !

Für SDL *fullscreen* statt in einem Fenster: Siehe auskommentierten Code in **sdlinterf.c** ! **SDL_X_SIZE** und **SDL_Y_SIZE** müssen *zusätzlich* auf die korrekte Fullscreen-Größe eingestellt werden!

c) Anzeige-Fehler (vor allem auf MacOS!):

Da es auf allen Rechnen mit MacOS sowie (je nach Grafikkarten-Treiber) auf manchen Windows- und Linux-Rechnern trotz korrekter Programme *seltsame Anzeige-Fehler* gab (beispielsweise gehen Teile der Anzeige verloren, oder schon gelöschte Pixel bleiben übrig), verwende ich jetzt in **sdlinterf.c** in der Funktion **sdlInit** den **Software-Renderer** (d.h. die CPU kopiert bei jedem Update die Inhalte des internen SDL-Puffers per C-Code in das SDL-Fenster auf dem Bildschirm), denn der hat bisher immer fehlerfrei funktioniert.

Wenn jemand die maximale Geschwindigkeit beim Bildschirm-Update benötigt, kann er versuchen, in **sdlinterf.c** stattdessen den Default-Renderer der jeweiligen Plattform zu nutzen (der entsprechende Aufruf von **SDL_CreateRenderer** ist auskommentiert), der meist OpenGL-Code auf dem Grafik-Prozessor nutzt, um Grafik-Daten zu kopieren.

d) Tastatur- und Maus-Input:

Wer meine SDL-Programme um *Tastatur- oder Maus-Input* erweitern will:
Hier gibt es massive *Einschränkungen* durch mein **sdlinterf** !
Näheres auf meiner Webseite: https://www.computerix.info/prog-sdl-input.html