

Programmieren für Physiker

Übung 11 im SS 2022

Achim Mildenberger | 08.07.2022

Grafik in C++: CImg, Kurzfassung

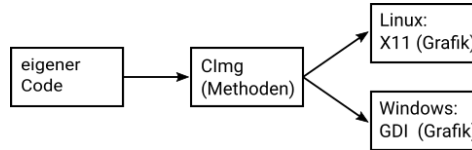
Allgemein:

- Grafik/Grafikbibliotheken sind nicht Bestandteil der Sprachdefinition von C++ oder den C++-Standardbibliotheken
- es existieren zahlreiche Grafikbibliotheken, teils in C++ oder anderen Sprachen geschrieben

CImg (<https://cimg.eu/>)

- bedient Grafik von
 - Linux: X11
 - Windows: gdi
- relativ klein: alles in einer .h-Datei

Bsp: cimg-example.cc



Grafik in C++: CImg, Intro

Grafikbefehle gehören nicht zum standardisierten Sprachumfang von C++. Dennoch ist es natürlich möglich, aus einem C++ Programm heraus Grafikelemente zu erzeugen und auszugeben. Dazu gibt es zahlreiche sogenannte "Bibliotheken", die Klassen oder Funktionen mehr oder weniger komfortabel zur Verfügung stellen. Eine relativ kleine und sehr praktische Bibliothek für Grafikausgaben und Bildbearbeitung ist [CImg](#).

Die Bibliothek ist in C++ geschrieben und hat zwei Vorteile:

- Mit dem gleichen Programmquelltext können sowohl unter Unix/Linux als auch unter Windows Grafiken ausgegeben werden.
- Es wird nur eine einzige (Quelltext-)Datei benötigt, die Datei `CImg.h`

Grafik in C++: Anleitung, Teil 1

- Um CImg zu verwenden, benötigen Sie lediglich die Datei `CImg.h` (mit im Zip-File auf der Kursseite). Diese enthält den Quelltext der Bibliothek in einer "Header-Datei". Verwenden Sie `#include "CImg.h"` in Ihrem Programm, um die Datei in Ihren Quelltext einzubeziehen. Dabei sollte CImg.h in Ihrem aktuellen Arbeitsverzeichnis sein.
- Compilieren: Bei der Compilation (genauer: beim "Linken") müssen die jeweiligen "fundamentalen" Grafikroutinen Ihres Betriebssystems eingebunden werden. (Diese sind ihrerseits ebenfalls Bibliotheken und werden von CImg aufgerufen. CImg ist also eigentlich nur ein "Bindeglied" zwischen Ihrem Programm und den Grafikbefehlen, die das Betriebssystem des Rechners mitbringt.) Welche Bibliotheken Sie hier angeben ist abhängig vom Betriebssystem, wo und wie Sie diese angeben ist abhängig von der Entwicklungsumgebung, die Sie verwenden.
Im Folgenden finden sich genauere Anweisungen.

Grafik in C++: Anleitung, Teil 2

- Compilieren Linux (Kommandozeile): `g++ IhrCode.cc -lX11 -lpthread`
(weitere Compileroptionen können natürlich verwendet werden). Falls Sie eine Entwicklungsumgebung einsetzen, sollte der Compileraufruf so konfiguriert werden, dass `-lX11 -lpthread` vorkommt. Da nun hierbei die Funktionsköpfe von X11 (fundamentale Grafikroutinen) des Linuxsystems verwendet werden, muss das Paket "libx11-dev" oder "xorg-x11-devel" (Name variiert etwas je nach Linux-Distribution) vorhanden sein.
Diese gleichen Compiler-Optionen gelten wahrscheinlich auch für Mac OS X.
- Compilieren Windows: Es muss die Bibliothek `gdi32` eingebunden werden. Unter Embarcadero-Dev-C++ ist der einfachste Weg, im Menüpunkt Tools/Compiler-Options (deutsch: Werkzeuge/Compiler Optionen) im Reiter Compiler unter "Add these commands to the linker command line" den Text `-lgdi32` einzutragen.
- Beispielprogramm: Mit dem Programm `cimg-example.cc` lernen Sie einige Befehle von `Cimg` kennen. Weitere Befehle und weiteres Know-How finden sich auf der [Webseite von Cimg in der Dokumentation](#)

Grafik in C++: Anleitung, Teil 3

Graphical User Interface (GUI):

Bei der Programmierung eines "Graphical User Interface" (GUI) bedient man sich üblicherweise eines [GUI-Toolkits/Widget-Toolkits](#).

Diese stellen in der Regel allerlei Grafikhilfsmittel, z.B. Schaltflächen, Menüfelder und Regler (Maus-/Touchbedienung) zur Verfügung. Im Allgemeinen sind sie deutlich mächtiger und umfangreicher als CImg, allerdings auch zum Einstieg etwas komplizierter.

Sehr weit verbreitet sind die plattformübergreifenden Toolkits [Qt](#) und [GTK+](#).