

## Inhalt

|  |   |    |
|--|---|----|
| 1.   | HMChart zum Programmierprojekt hinzufügen.....        | 1  |
| 2.   | Beispiele.....  | 3  |
| 2.1.   | Einfache Liniengrafik.....                            | 3  |
| 2.2.   | Punkte und Linien.....                                | 4  |
| 2.3.   | Einstellung der Achsenbeschriftung.....               | 5  |
| 2.4.   | Manuelle Achsenskalierung.....                        | 6  |
| 2.5.   | Animation aus mehreren Einzelbildern.....             | 7  |
| 2.6.   | Kurven oder Punktsequenzen in einem Zug zeichnen..... | 8  |
| 2.7.   | Hintergrundfarbe einstellen.....                      | 9  |
| 2.8.   | Programmierung in C+.....                             | 10 |
| 3.   | Funktionsübersicht.....                               | 11 |
| 3.1.   | Zusätzliche Funktionen in C++.....                    | 13 |
| 4.   | Hinweise.....   | 14 |
| 5.   | Kontakt, Lizenz.....                                  | 15 |
| Anhang A: Charts in Benutzeroberflächen integrieren..... |   | 16 |

## 1. HMChart zum Programmierprojekt hinzufügen

HMChart ermöglicht die Programmierung einfacher 2D-Charts in den Programmiersprachen C und C++ unter Microsoft Windows und dem X Window System (X11). Die X11-Version kann sowohl unter Linux als auch unter Mac OS X genutzt werden. Die Charts werden in separaten Fenstern angezeigt, der Einsatz von HMChart ist daher – gerade auch – aus Konsolenanwendungen heraus möglich.

### Microsoft Visual Studio (Betriebssystem: Microsoft Windows)

Falls die Entwicklungsumgebung Microsoft Visual Studio genutzt wird, sind lediglich die zwei Dateien **chart.h** und **chart-windows.c** zum Projekt hinzuzufügen. Die folgende Abbildung zeigt Microsoft Visual Studio 2015 mit einem geöffneten Projekt. Auf der linken Seite ist der Projektmappen-Explorer sichtbar, in dem die beiden genannten Dateien aufgelistet sind.

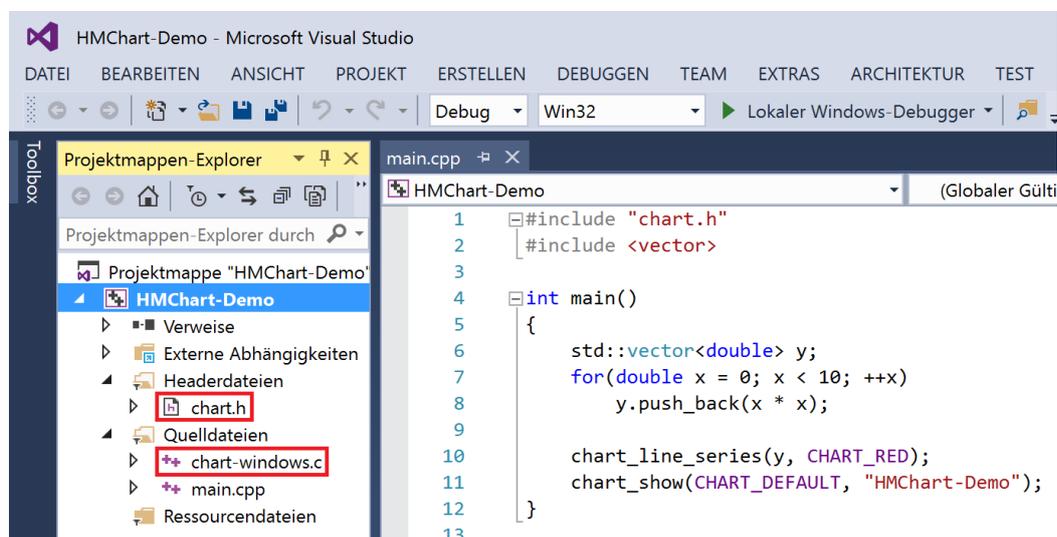


Abbildung 1 – Microsoft Windows, Benutzeroberfläche von Microsoft Visual Studio 2015

## Qt Creator (Betriebssysteme: Microsoft Windows, Linux, Mac OS X)

Falls die Entwicklungsumgebung Qt Creator genutzt wird, sind zunächst die zwei Dateien **chart.h** und **chart-windows.c** (Microsoft Windows) bzw. **chart.h** und **chart-x11.c** (Mac OS X und Linux) zum Projekt hinzuzufügen. Anschließend müssen die folgenden Einträge in der Projektdatei (.pro-Datei) ergänzt werden, siehe auch Abbildung 2:

- Microsoft Windows: `LIBS += -luser32 -lgdi32 -lcomdlg32`
- Mac OS X: `INCLUDEPATH += /opt/X11/include/`  
`LIBS += -L/opt/X11/lib/ -lX11 -lm`
- Linux: `LIBS += -lX11 -lm`

**Bitte beachten: HMChart benötigt auf Mac-OS-X-Rechnern das X Window System (X11). Zusätzlich zur Entwicklungsumgebung (Qt Creator, Xcode...) ist dazu X11/XQuartz zu installieren. Es kann von der folgenden Webseite heruntergeladen werden: [www.xquartz.org/](http://www.xquartz.org/)**

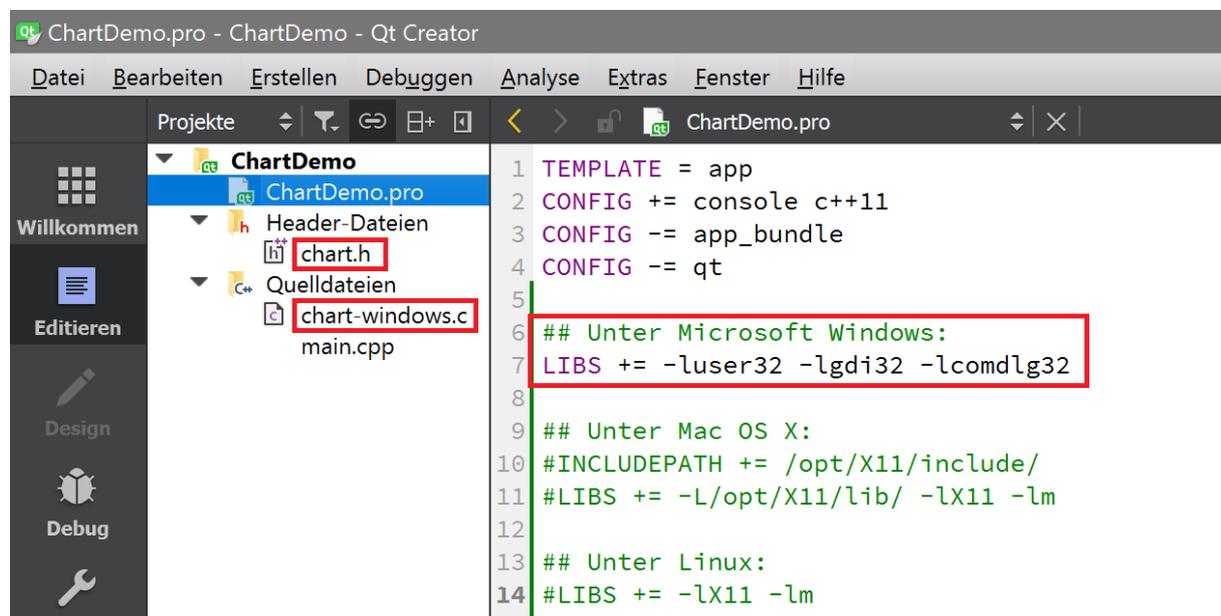


Abbildung 2 – Qt Creator mit einer geöffneten Chart-Projektdatei

## XCode (Betriebssystem: Mac OS X)

Falls die Entwicklungsumgebung Xcode (Mac OS X) genutzt wird, sind zunächst die zwei Dateien **chart.h** und **chart-x11.c** zum Projekt hinzuzufügen. Anschließend müssen auf der Seite „Build Settings (All)“ die folgenden Einstellungen hinzugefügt werden:

- Other Linker Flags: `-L/opt/X11/lib/ -lX11 -lm`
- Header Search Paths: `/opt/X11/include/`

**Bitte beachten: HMChart benötigt auf Mac-OS-X-Rechnern das X Window System (X11). Zusätzlich zur Entwicklungsumgebung (Qt Creator, Xcode...) ist dazu X11/XQuartz zu installieren. Es kann von der folgenden Webseite heruntergeladen werden: [www.xquartz.org/](http://www.xquartz.org/)**

## Erstellen von HMChart-Projekten über die Kommandozeile

Die Programmerstellung unter Linux und Mac OS X geschieht häufig direkt über die Kommandozeile, also ohne Verwendung einer grafischen Entwicklungsumgebung. Unter Linux können Compiler und Linker mit den folgenden Kommandos aufgerufen werden:

```
gcc -Wall -c my-own-program.c
gcc -Wall -c chart-x11.c
gcc -Wall -o my-own-program *.o -lX11 -lm
```

Unter Mac OS X:

```
gcc -I/opt/X11/Include/ -Wall -c my-own-program.c
gcc -I/opt/X11/Include/ -Wall -c chart-x11.c
gcc -L/opt/X11/lib/ -Wall -o my-own-program *.o -lX11 -lm
```

**Bitte beachten:** HMChart benötigt auf Mac-OS-X-Rechnern das X Window System (X11). Zusätzlich zur Entwicklungsumgebung (Qt Creator, Xcode...) ist dazu X11/XQuartz zu installieren. Es kann von der folgenden Webseite heruntergeladen werden: [www.xquartz.org/](http://www.xquartz.org/)

## 2. Beispiele

### 2.1. Einfache Liniengrafik

```
/* Folgende Datei zum Projekt hinzufügen / Add this file to project: */
/* --> chart-windows.c (Microsoft Windows) */
/* --> chart-x11.c (für Linux u. Mac OS X) */
#include "chart.h"

int main(void)
{
    /* Haus */
    chart_moveto(2, 0);
    chart_lineto(2, 2, CHART_BLUE); chart_lineto(1, 3, CHART_BLUE);
    chart_lineto(0, 2, CHART_BLUE); chart_lineto(2, 2, CHART_BLUE);
    chart_lineto(0, 0, CHART_BLUE); chart_lineto(0, 2, CHART_BLUE);
    chart_lineto(2, 0, CHART_BLUE); chart_lineto(0, 0, CHART_BLUE);

    /* Garten */
    chart_moveto(-1.0, -0.05); chart_lineto( 3.0, -0.05, CHART_GREEN);
    chart_moveto(-1.5, -0.10); chart_lineto( 3.5, -0.10, CHART_GREEN);

    /* Chart anzeigen */
    chart_show(CHART_NOBACKGROUND | CHART_NOGRID, "Haus vom Nikolaus");
    return 0;
}
```

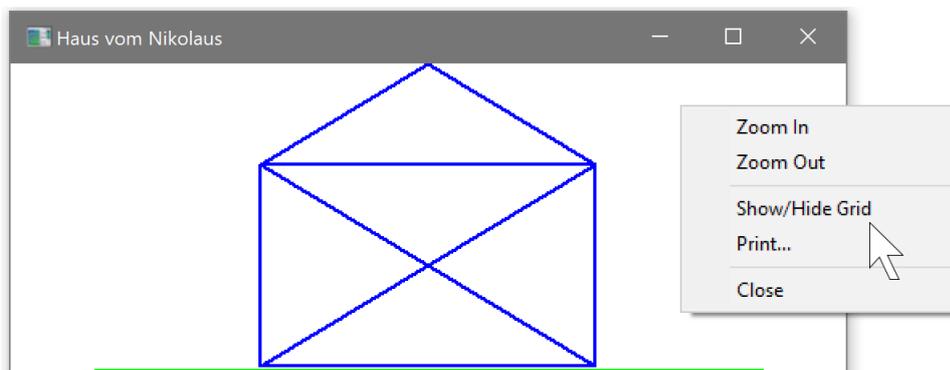


Abbildung 3 – Einfache Liniengrafik

## 2.2. Punkte und Linien

```
/* Folgende Datei zum Projekt hinzufügen / Add this file to project: */
/* --> chart-windows.c (Microsoft Windows) */
/* --> chart-x11.c (für Linux u. Mac OS X) */

#include "chart.h"
#include "math.h"

int main(void)
{
    double x, y;

    /* Startposition fuer Sinuskurve */
    x = -5; y = sin(x); chart_moveto(x, y);

    /* Sinuskurve zeichnen */
    for(x = -5; x < 5; x += 0.1)
    {
        y = sin(x);
        chart_lineto(x, y, CHART_BLUE);
    }

    /* Startposition fuer Kosinuskurve */
    x = -5; y = cos(x); chart_moveto(x, y);

    /* Kosinuskurve zeichnen */
    for(x = -5; x < 5; x += 0.1)
    {
        y = cos(x);
        chart_lineto(x, y, CHART_GREEN);
    }

    /* Verlauf von sin(x) * cos(x) mit Punkten zeichnen */
    for(x = -5; x < 5; x += 0.1)
    {
        y = sin(x) * cos(x);
        chart_point(x, y, CHART_RED);
    }

    /* Chart anzeigen */
    chart_show(CHART_DEFAULT, "Sinus, Kosinus");
    return 0;
}
```

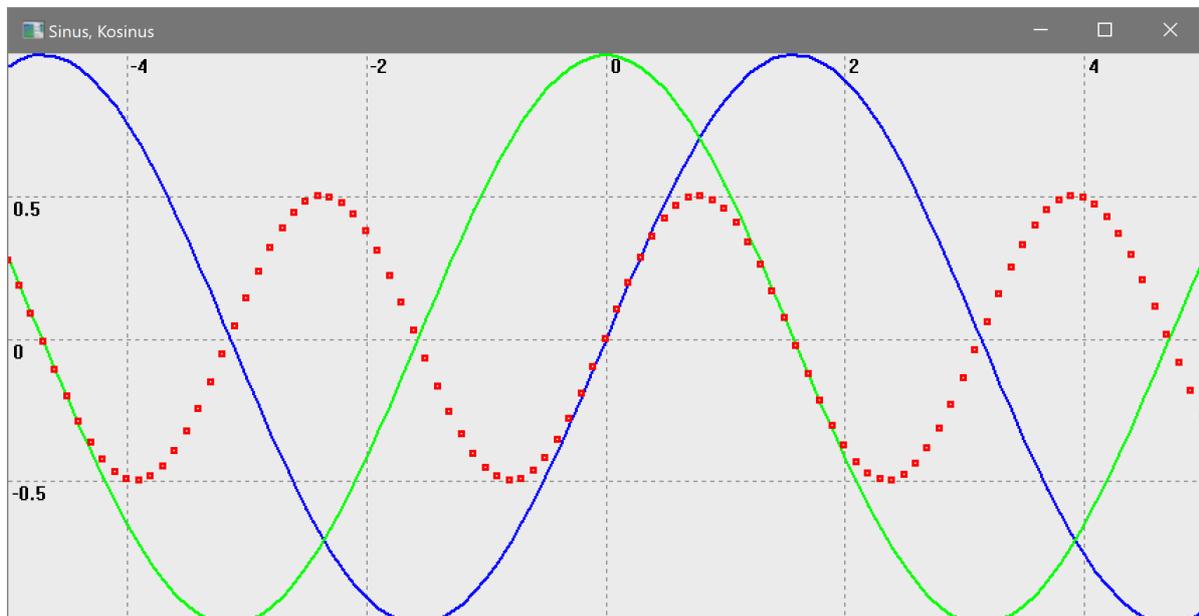


Abbildung 4 – Punkte und Linien

## 2.3. Einstellung der Achsenbeschriftung

```
/* Folgende Datei zum Projekt hinzufügen / Add this file to project: */
/* --> chart-windows.c (Microsoft Windows) */
/* --> chart-x11.c (für Linux u. Mac OS X) */

#include "chart.h"
#include "math.h"

int main(void)
{
    double t, u;
    double omega = 2 * 3.1416 * 50;
    double effektivwert = 230;

    /* Sinuskurve zeichnen */
    for(t = 0; t < 100; t += 0.1)
    {
        u = sqrt(2) * effektivwert * sin(omega * t / 1000.0);
        chart_lineto(t, u, CHART_RED);
    }

    /* Format fuer x- und y-Achse uebergeben (wie bei printf) */
    chart_axis_fmt("%.1f ms", "%.1f Volt");

    /* Chart darstellen */
    chart_show(CHART_DEFAULT, "Spannungsverlauf in der Steckdose");
    return 0;
}
```

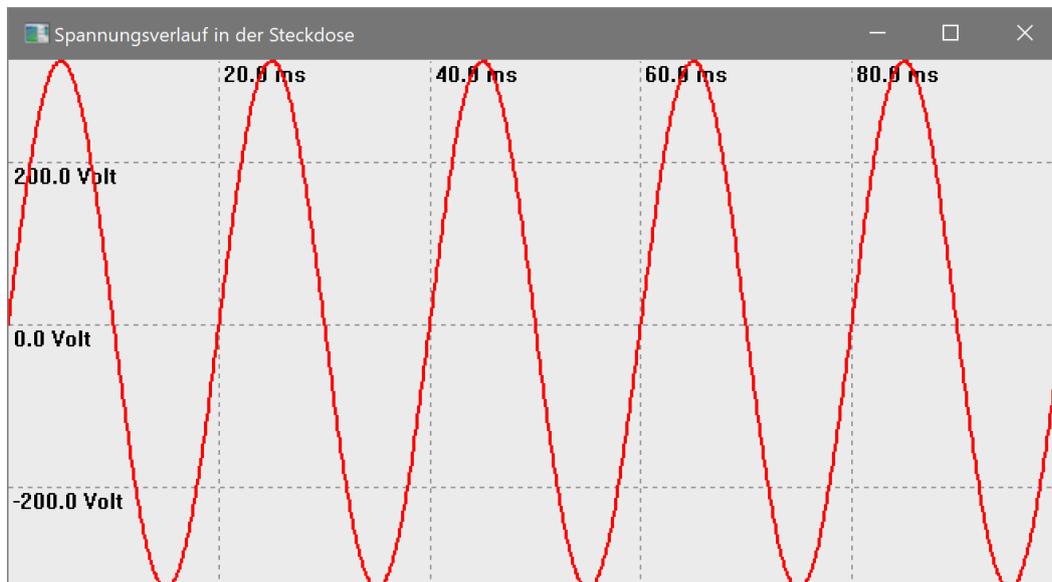


Abbildung 5 – Einstellung der Achsenbeschriftung

## 2.4. Manuelle Achsenskalierung

```
/* Folgende Datei zum Projekt hinzufügen / Add this file to project: */
/* --> chart-windows.c (Microsoft Windows) */
/* --> chart-x11.c (für Linux u. Mac OS X) */

#include "chart.h"
#include <math.h>

int main(void)
{
    /* Funktion  $y = x * \sin(1.0 / x)$  grafisch darstellen */
    double x = -1.0, y = x * sin(1.0 / x);
    chart_moveto(x, y);
    for(x = -1.0; x < 1.0; x += 0.00002)
    {
        y = x * sin(1.0 / x);
        chart_lineto(x, y, CHART_BLUE);
    }

    /* Der x-Bereich hat eigentlich eine Ausdehnung von -1.0 ... 1.0 */
    chart_axis_xlimit(-0.05, 0.15);

    /* Der y-Bereich hat eigentlich eine Ausdehnung von -0.842 ... 0.842 */
    chart_axis_ylimit(-0.075, 0.075);

    /* Manuelle Skalierung mittels CHART_XLIMIT, CHART_YLIMIT aktivieren */
    chart_show(CHART_XLIMIT | CHART_YLIMIT, "Manuelle Achsenskalierung");
    return 0;
}
```

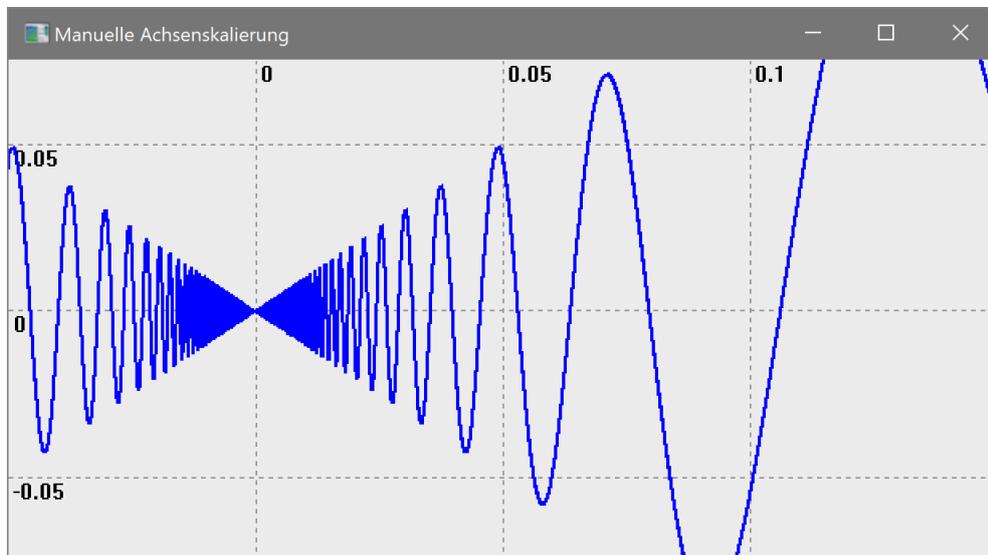


Abbildung 6 – Manuelle Achsenskalierung

## 2.5. Animation aus mehreren Einzelbildern

```
/* Folgende Datei zum Projekt hinzufügen / Add this file to project: */
/* --> chart-windows.c (Microsoft Windows) */
/* --> chart-x11.c (für Linux u. Mac OS X) */

#include "chart.h"

int main(void)
{
    /* Erstes Einzelbild: blaues Viereck */
    chart_lineto(1.5, 1.5, CHART_BLUE);
    chart_lineto(3.0, 0.0, CHART_BLUE);
    chart_lineto(1.5, -1.5, CHART_BLUE);
    chart_lineto(0.0, 0.0, CHART_BLUE);
    chart_end_of_frame();

    /* Zweites Einzelbild: rotes Dreieck */
    chart_lineto(1.5, 1.5, CHART_RED);
    chart_lineto(1.5, -1.5, CHART_RED);
    chart_lineto(0.0, 0.0, CHART_RED);
    chart_end_of_frame();

    /* Jedes Einzelbild wird 500 msec dargestellt */
    chart_frame_timer(500);
    chart_show(CHART_DEFAULT, "Animation");
    return 0;
}
```

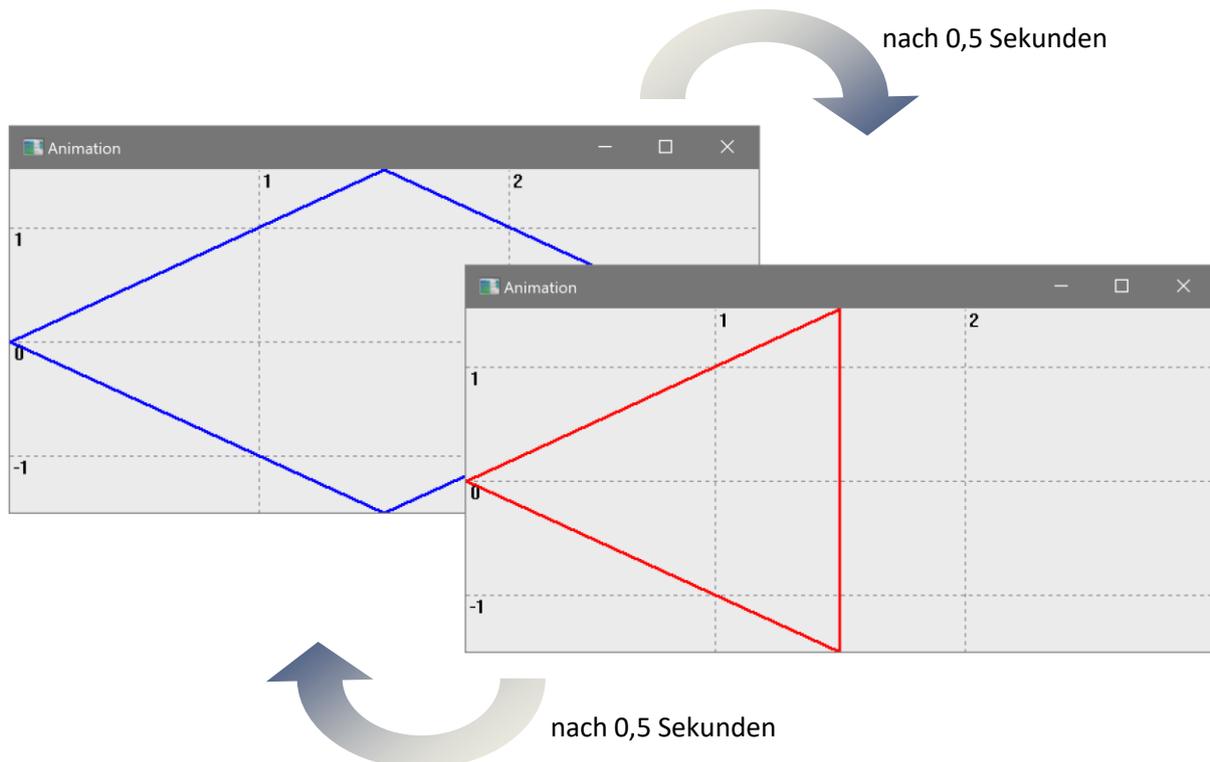


Abbildung 7 – Animation aus mehreren Einzelbildern

## 2.6. Kurven oder Punktsequenzen in einem Zug zeichnen

```
/* Folgende Datei zum Projekt hinzufügen / Add this file to project: */
/* --> chart-windows.c (Microsoft Windows) */
/* --> chart-x11.c (für Linux u. Mac OS X) */

#include "chart.h"
#include <math.h>
#define DELTA 0.1
#define DIM 50

int main(void)
{
    int i;
    double x1[DIM], y1[DIM], y2[DIM];
    double z[] = { -2, -1, 0, 1, 2, 3 };

    for(i = 0; i < DIM; ++i)
    {
        double x = i * DELTA;
        x1[i] = x; y1[i] = x * sin(x); y2[i] = x * cos(x);
    }

    /* Linienzug mit "DIM" Koordinaten zeichnen */
    chart_line_series(DIM, x1, y1, CHART_GREEN);

    /* Punktsequenz mit "DIM" Punkten zeichnen */
    chart_point_series(DIM, x1, y2, CHART_RED);

    /* Die x-Koordinaten können entfallen, in diesem
     * Fall werden sie zu 0, 1, 2, 3 usw. gesetzt. */
    chart_point_series1(6, z, CHART_BLUE);

    /* Chart darstellen */
    chart_show(CHART_DEFAULT | CHART_NOBACKGROUND,
               "Kurven in einem Zug zeichnen");
    return 0;
}
```

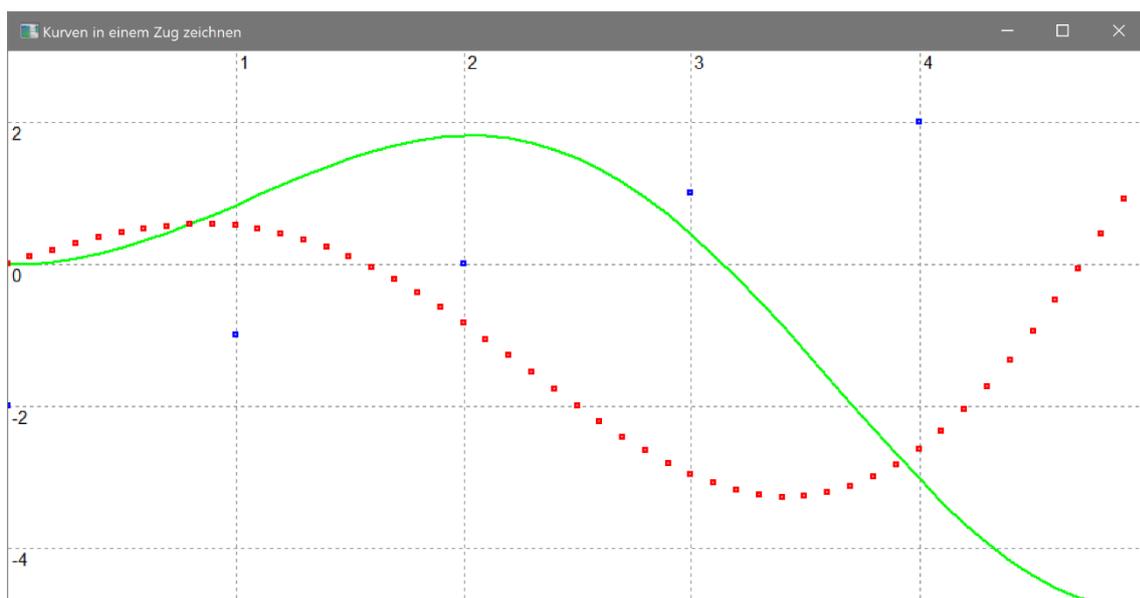


Abbildung 8 – Kurven in einem Zug zeichnen

## 2.7. Hintergrundfarbe einstellen

```
// Folgende Datei zum Projekt hinzufügen / Add this file to project:
//      --> chart-windows.c (Microsoft Windows)
//      --> chart-x11.c (für Linux u. Mac OS X)

#include "chart.h"
#include <stdlib.h>

int main(void)
{
    /* Bunte Punkte auf schwarzem Hintergrund */
    chart_clear();
    chart_background(0, 0, 0);

    for(int i = 0; i < 5000; ++i)
    {
        double x = 2.0 * rand() / RAND_MAX - 1.0;
        double y = 2.0 * rand() / RAND_MAX - 1.0;
        int col = rand() % 5;
        if(x * x + y * y < 1) chart_point(x, y, col);
    }

    chart_show(CHART_NOGRID, "Bunte Punkte auf schwarzem Hintergrund");
    return 0;
}
```

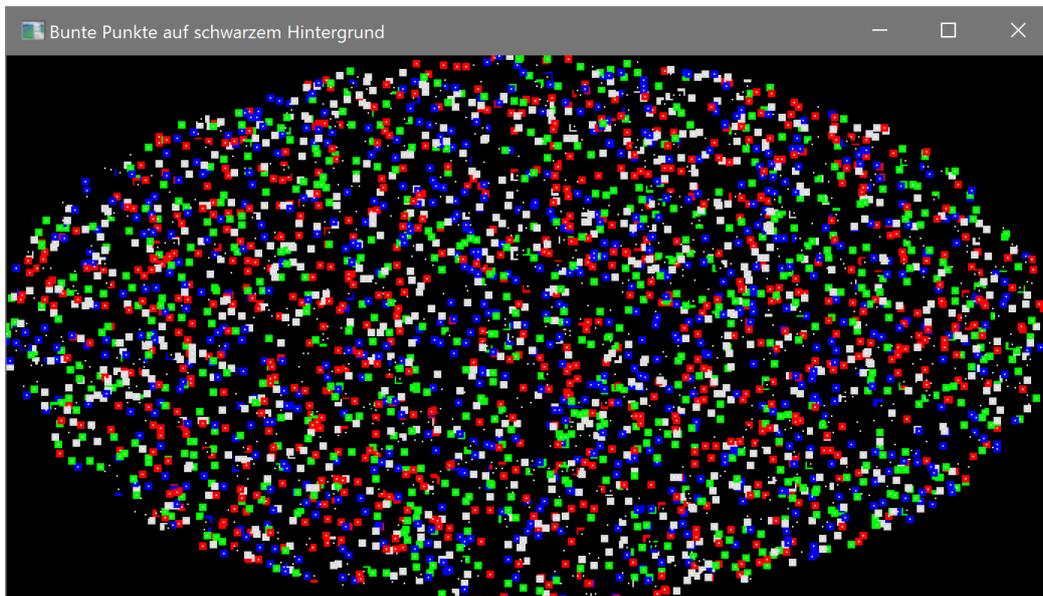


Abbildung 9 – Hintergrundfarbe einstellen

## 2.8. Programmierung in C++

```
// Folgende Datei zum Projekt hinzufügen / Add this file to project:
// --> chart-windows.c (Microsoft Windows)
// --> chart-x11.c (für Linux u. Mac OS X)

#include "chart.h"
#include <math.h>
#include <vector>
using namespace std;

int main()
{
    vector<double> x_vec; // Vektor mit x-Koordinaten
    vector<double> y1_vec; // Vektor mit Sinuswerten
    vector<double> y2_vec; // Vektor mit Kosinuswerten
    for(double x = -5.0; x < 5.0; x += 0.1)
    {
        double y1 = sin(x);
        double y2 = cos(x);
        x_vec.push_back(x);
        y1_vec.push_back(y1);
        y2_vec.push_back(y2);
    }

    // C++-Container können direkt an chart_line_series()
    // und chart_point_series() übergeben werden!
    chart_line_series(x_vec, y1_vec, CHART_RED);
    chart_point_series(x_vec, y2_vec, CHART_BLUE);
    chart_show(CHART_DEFAULT, "C++-Demo");
}
```

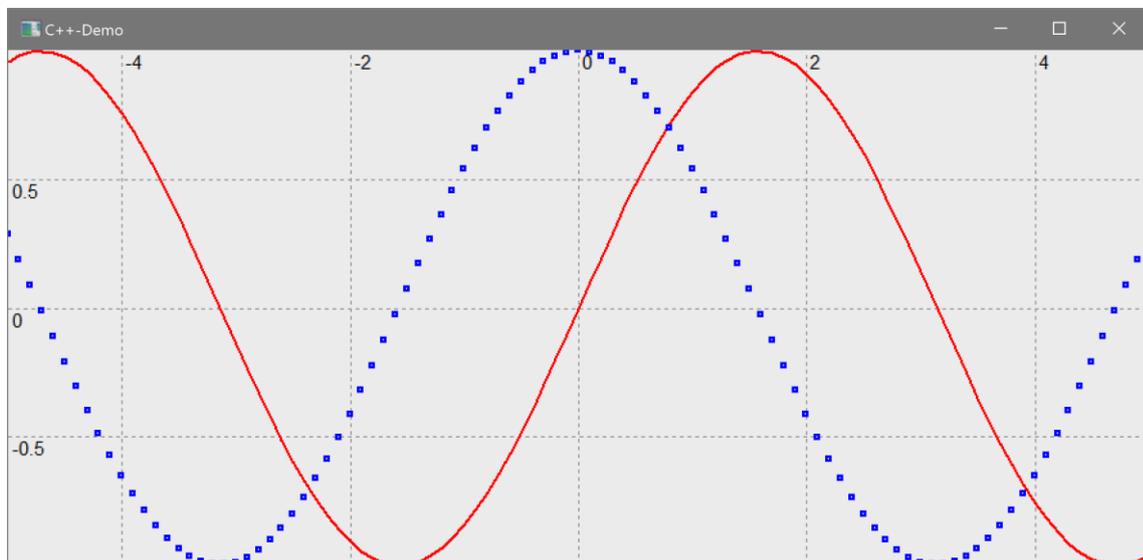


Abbildung 10 – Programmierung in C++

### 3. Funktionsübersicht

- `void chart_moveto(double x, double y);`  
Aktuelle Ausgabeposition setzen.
- `void chart_lineto(double x, double y, int color);`  
Linie von der aktuellen Ausgabeposition zum angegebenen Punkt zeichnen. Für den Parameter "color" sind folgende Werte möglich: CHART\_BLACK, CHART\_RED, CHART\_GREEN, CHART\_BLUE und CHART\_GRAY.
- `void chart_point(double x, double y, int color);`  
Markierung am angegebenen Punkt ausgeben. Für den Parameter "color" sind folgende Werte möglich: CHART\_BLACK, CHART\_RED, CHART\_GREEN, CHART\_BLUE und CHART\_GRAY.
- `void chart_clear(void);`  
Alle Linien und Punkte löschen, Ausgabeposition auf (0; 0) setzen.
- `void chart_axis_fmt(const char *xfmt, const char *yfmt);`  
Formatstrings zur Beschriftung der x- und y-Achsen einstellen, Default = "%g". Die Formatstrings sehen aus wie bei der Bibliotheksfunktion "printf" (Ein Beispielprogramm findet sich im Abschnitt "Einstellung der Achsenbeschriftung" dieser Dokumentation).
- `void chart_axis_xlimit(double xmin, double xmax);`  
Manuelle Achsenskalierung: Minimal-/Maximalwerte für die x-Achse angeben. Um die manuelle Achsenskalierung für die x-Achse zu aktivieren, muss beim Aufruf von "chart\_show" die Option "CHART\_XLIMIT" angegeben sein.
- `void chart_axis_ylimit(double ymin, double ymax);`  
Manuelle Achsenskalierung: Minimal-/Maximalwerte für die y-Achse angeben. Um die manuelle Achsenskalierung für die y-Achse zu aktivieren, muss beim Aufruf von "chart\_show" die Option "CHART\_YLIMIT" angegeben sein.
- `void chart_show(int options, const char *title);`  
Chart anzeigen und warten, bis der Anwender das Fenster schließt. Für den Parameter "options" sind die Werte CHART\_DEFAULT, CHART\_XLIMIT, CHART\_YLIMIT, CHART\_NOBACKGROUND, CHART\_NOGRID, CHART\_XZOOM\_OFF, CHART\_YZOOM\_OFF und CHART\_NOCLEAR erlaubt. Mehrere Optionen können mit dem |-Operator kombiniert werden:
  - So wird durch (CHART\_NOGRID | CHART\_NOBACKGROUND) ein Chart ohne Hintergrund und Koordinatensystem angezeigt.
  - Mittels CHART\_XLIMIT und CHART\_YLIMIT kann die automatische Achsenskalierung deaktiviert werden; der darstellbare Bereich wird dann mit `chart_axis_xlimit()` und `chart_axis_ylimit()` eingestellt.
  - Mittels CHART\_XZOOM\_OFF und CHART\_YZOOM\_OFF kann die Zoom-Funktionalität des Charts in X- oder Y-Richtung oder auch vollständig deaktiviert werden. Zum vollständigen Deaktivieren ist (CHART\_XZOOM\_OFF | CHART\_YZOOM\_OFF) zu übergeben.
  - Wenn `chart_show()` mit der Option CHART\_NOCLEAR aufgerufen wird, werden die Chart-Objekte (Linien, Punkte usw.) nach dem Schließen des Charts nicht gelöscht. Zum "manuellen Löschen" muss dann bei Bedarf die Funktion `chart_clear()` aufgerufen werden.
- `void chart_line_series(size_t n, const double *x, const double *y, int color);`  
Eine komplette Kurve aus "n" aufeinanderfolgenden x- bzw. y-Koordinaten wird in einem Zug dargestellt. Die aktuelle Ausgabeposition befindet sich anschließend am letzten Stützpunkt der Kurve. Für den Parameter "color" sind die folgenden Werte möglich: CHART\_BLACK, CHART\_RED, CHART\_GREEN, CHART\_BLUE und CHART\_GRAY. (Der Aufrufer dieser Funktion muss sicherstellen, dass die Parameter "x" und "y" jeweils auf den Beginn eines Vektors mit "n" double-Werten zeigen!)

- `void chart_point_series(size_t n, const double *x, const double *y, int col);`  
Eine Punktsequenz aus "n" aufeinanderfolgenden x- bzw. y-Koordinaten wird in einem Zug dargestellt. Die aktuelle Ausgabeposition wird durch den Aufruf dieser Funktion nicht verändert. Für den Parameter "col" sind die folgenden Werte möglich: CHART\_BLACK, CHART\_RED, CHART\_GREEN, CHART\_BLUE und CHART\_GRAY. (Der Aufrufer muss sicherstellen, dass die Parameter "x" und "y" jeweils auf den Beginn eines Vektors mit "n" double-Werten zeigen!)
- `void chart_line_series1(size_t n, const double *y, int color);`  
Variante von `chart_line_series()`, es wird allerdings nur eine Liste von y-Koordinaten übergeben. Die x-Koordinaten werden automatisch vergeben: 0, 1, 2, 3 usw...
- `void chart_point_series1(size_t n, const double *y, int color);`  
Variante von `chart_point_series()`, es wird allerdings nur eine Liste von y-Koordinaten übergeben. Die x-Koordinaten werden automatisch vergeben: 0, 1, 2, 3 usw...
- `void chart_frame_timer(int msec);`  
Bei einem animierten Chart kann durch `chart_frame_timer()` eingestellt werden, wie lange jedes Einzelbild (sog. Frame) auf dem Bildschirm angezeigt wird, siehe auch: `chart_end_of_frame()`. Minimal können 100 msec, maximal 10000 msec eingestellt werden.
- `void chart_end_of_frame(void);`  
Es wird ein animiertes Chart aufgebaut, das aus mehreren Einzelbildern (sog. Frames) besteht. Der Aufruf von `chart_end_of_frame()` bewirkt, dass ein weiteres Einzelbild zum Chart hinzugefügt wird – nachfolgende Aufrufe von `chart_point()`, `chart_moveto()`, `chart_lineto()` beziehen sich auf das neue Einzelbild. Die Ausgabeposition wird von `chart_end_of_frame()` auf (0; 0) zurückgesetzt. Beispiel: Soll eine Animation erstellt werden, die aus lediglich zwei Einzelbildern besteht, so werden zunächst die Punkte und Linien des ersten Einzelbildes mittels `chart_point()` usw. definiert. Es folgt der Aufruf von `chart_end_of_frame()`. Nun werden die Punkte und Linien des zweiten Einzelbildes definiert und schließlich `chart_show()` zur Anzeige auf dem Bildschirm aufgerufen. Beide Einzelbilder werden automatisch abwechselnd angezeigt, die Geschwindigkeit der Animation kann mittels `chart_frame_timer()` eingestellt werden.
- `void chart_grid_threshold(int width, int height);`  
Unterschreitet die Breite bzw. Höhe des Charts eine bestimmte Grenze, wird die Zahl der vertikalen bzw. horizontalen Gitterlinien verringert. Mittels `chart_grid_threshold()` können diese Grenzen eingestellt werden.
- `void chart_background(unsigned char r, unsigned char g, unsigned char b);`  
Hintergrundfarbe einstellen (Standardfarbe = Hellgrau). In den Parametern r, g und b werden die Rot-, Grün- und Blau-Anteile der gewünschten Farbe übergeben (jeweils im Bereich 0...255).

### 3.1. Zusätzliche Funktionen in C++

- ```
template <class Vector>
void chart_point_series(const Vector& xvec, const Vector& yvec, int color);
```
- ```
template <class Vector>
void chart_point_series(const Vector& yvec, int color);
```

Eine Punktsequenz aus aufeinanderfolgenden x- bzw. y-Koordinaten wird in einem Zug dargestellt. Die x- bzw. y-Koordinaten sind in C++-Containerklassen gespeichert. Die aktuelle Ausgabeposition wird durch den Aufruf dieser Funktion nicht verändert. Für den Parameter "color" sind die folgenden Werte möglich: CHART\_BLACK, CHART\_RED, CHART\_GREEN, CHART\_BLUE und CHART\_GRAY. Die x-Koordinaten müssen nicht angegeben werden, siehe chart\_point\_series1().

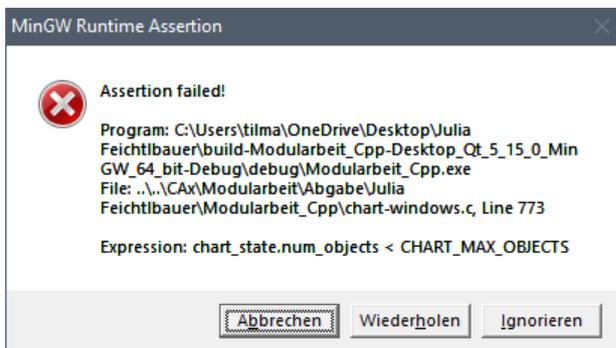
- ```
template <class Vector>
void chart_line_series(const Vector& xvec, const Vector& yvec, int color);
```
- ```
template <class Vector>
void chart_line_series(const Vector& yvec, int color);
```

Eine komplette Kurve aus aufeinanderfolgenden x- bzw. y-Koordinaten wird in einem Zug dargestellt. Die x- bzw. y-Koordinaten sind in C++-Containerklassen gespeichert. Die aktuelle Ausgabeposition wird durch den Aufruf dieser Funktion nicht verändert. Für den Parameter "color" sind die folgenden Werte möglich: CHART\_BLACK, CHART\_RED, CHART\_GREEN, CHART\_BLUE und CHART\_GRAY. Die x-Koordinaten müssen nicht angegeben werden, siehe chart\_line\_series1().

## 4. Hinweise

- In der **Windows-Version** öffnet sich durch Klick mit der rechten Maustaste ein Popup-Menü. Hier stehen Menüpunkte zum Vergrößern („Zoom In“) und Verkleinern („Zoom Out“), zum Ein- und Ausblenden der Gitterlinien und Achsenbeschriftungen („Show/Hide Grid“), zum Drucken („Print...“) und zum Schließen des Chart-Fensters zur Verfügung. Zum Vergrößern der Ansicht kann auch mit der linken Maustaste direkt an die gewünschte Position im Chart geklickt werden.
- In der **X11-Version** (unter **Linux** und **Mac OS X**) existiert kein Popup-Menü. Stattdessen können die folgenden Funktionen direkt per Tastendruck gewählt werden: Gitterlinien ein-/ausblenden (Taste „g“), Chart schließen (Taste „q“). Die Betätigung der linken Maustaste führt wie bei der Windows-Version zur Vergrößerung der Ansicht („Zoom In“), die Betätigung der rechten Maustaste stellt dagegen direkt die ursprüngliche Ansicht wieder her („Zoom Out“).
- Die Programmierschnittstelle ist bewusst einfach gehalten. So sind zum Einsatz von HMChart keine Zeiger oder „Chart-Handles“ erforderlich. Charts müssen auch nicht „angelegt“ oder nach der Verwendung wieder „freigegeben“ werden. Dies wurde durch die Speicherung des aktuellen Chart-Zustands (inkl. aller dargestellten Linien und Punkte) in statischen, globalen Variablen erreicht, wodurch sich folgende Einschränkungen ergeben:
  - Die gleichzeitige Anzeige mehrerer Charts innerhalb einer Applikation ist nicht möglich,
  - HMChart ist nicht threadsicher.
- In einem Chart können **bis zu 100.000 Objekte** (Punkte, Linien usw.) gleichzeitig dargestellt werden. Reicht dies nicht aus, so kann die maximale Objekt-Anzahl durch Definition der symbolischen Konstante `CHART_MAX_OBJECTS` auf einen höheren Wert gesetzt werden (siehe Quell-datei **chart-windows.c** bzw. **chart-x11.c**, ca. Zeile 55).

Wenn versucht wird, mehr als `CHART_MAX_OBJECTS` Objekte zum Chart hinzuzufügen, dann erscheint die folgende Fehlermeldung:



## 5. Kontakt, Lizenz



Tilman Küpper  
[tilman.kuepper@hm.edu](mailto:tilman.kuepper@hm.edu)

Hochschule München  
Fakultät für Maschinenbau, Fahrzeugtechnik, Flugzeugtechnik  
Dachauer Straße 98 b  
D-80335 München

<http://kuepper.userweb.mwn.de/>

Boost Software License - Version 1.0 - August 17th, 2003

Permission is hereby granted, free of charge, to any person or organization obtaining a copy of the software and accompanying documentation covered by this license (the "Software") to use, reproduce, display, distribute, execute, and transmit the Software, and to prepare derivative works of the Software, and to permit third-parties to whom the Software is furnished to do so, all subject to the following:

The copyright notices in the Software and this entire statement, including the above license grant, this restriction and the following disclaimer, must be included in all copies of the Software, in whole or in part, and all derivative works of the Software, unless such copies or derivative works are solely in the form of machine-executable object code generated by a source language processor.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR ANYONE DISTRIBUTING THE SOFTWARE BE LIABLE FOR ANY DAMAGES OR OTHER LIABILITY, WHETHER IN CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Anhang A: Charts in Benutzeroberflächen integrieren

Unter dem Betriebssystem Microsoft Windows ist es möglich, Charts direkt in existierende Benutzeroberflächen zu integrieren anstatt sie – wie sonst üblich – in separaten Fenstern zu öffnen. Dazu ist allerdings etwas „Handarbeit“ notwendig.

Beispielhaft wird in diesem Anhang gezeigt, wie eine **MFC-Dialogapplikation mit integriertem Chart** erstellt werden kann. Als Entwicklungsumgebung kommt Microsoft Visual Studio 2015 zum Einsatz.

### 1. Schritt: Neue MFC-Anwendung anlegen

Mit dem MFC-Anwendungs-Assistenten wird eine neue MFC-Anwendung angelegt. Der Name der Applikation lautet **ChartDemo**, der Anwendungstyp ist: **Auf Dialogfeldern basierend**. Alle übrigen Einstellungen im Anwendungs-Assistenten bleiben unverändert.

### 2. Schritt: HMChart-Klasse zum Projekt hinzufügen

Zunächst werden – wie bereits in der Einleitung gezeigt – die beiden Dateien **chart.h** und **chart-windows.c** zum Projekt hinzugefügt. Da HMChart in der Programmiersprache C implementiert wurde, können vorkompilierte (C++)-Headerdateien hier nicht verwendet werden. Sie werden daher abgeschaltet: Projekt → Eigenschaften → C/C++ → Vorkompilierte Header → Vorkompilierte Header nicht verwenden.

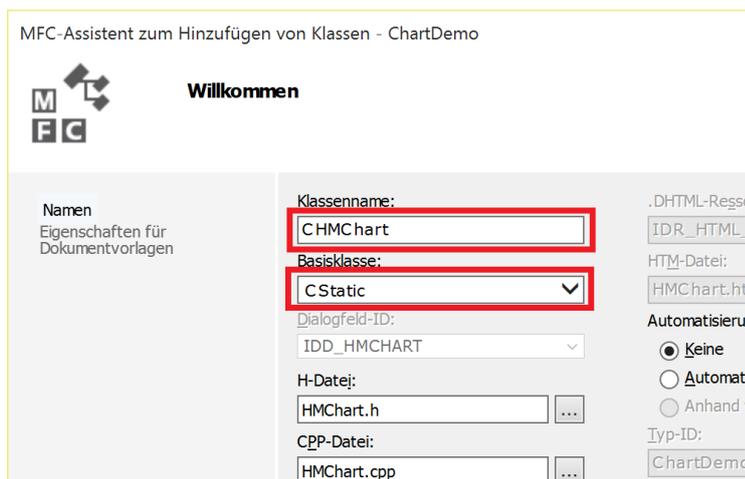


Abbildung A1 – MFC-Klasse CHMChart erzeugen

Nun wird der Klassenassistent geöffnet (Strg+Umschalt+x) und eine neue MFC-Klasse **CHMChart** erzeugt (Klasse hinzufügen → MFC-Klasse), welche später im Dialogfenster die Darstellung des Charts übernimmt, siehe Abbildung A1. Als Basisklasse wird **CStatic** ausgewählt.

Vor dem Schließen des Klassenassistenten überprüfen Sie nochmals die Dateinamen: Die neue Klasse soll in den Dateien **HMChart.h** bzw. **HMChart.cpp** abgespeichert werden.

### 3. Schritt: Meldungs-Handler zur Klasse CHMChart hinzufügen

Der Klassenassistent wird ein weiteres Mal geöffnet, um vier Meldungs-Handler für die Windows-Meldungen **WM\_PAINT**, **WM\_CLOSE**, **WM\_LBUTTONDOWN** und **WM\_RBUTTONDOWN** zur neu erzeugten Klasse **CHMChart** hinzuzufügen, siehe Abbildung A2.

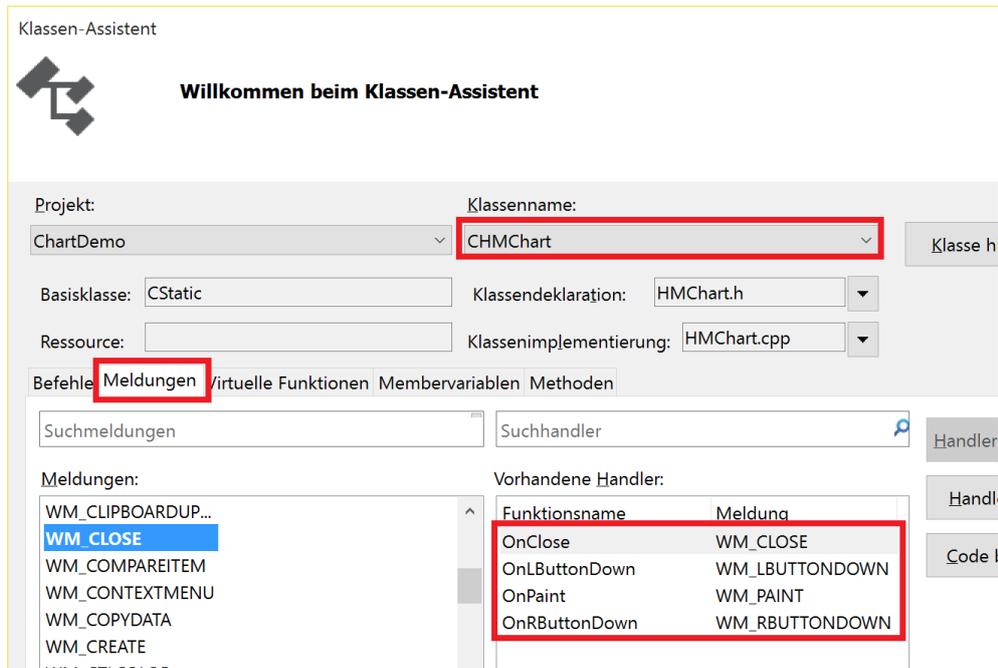


Abbildung A2 – Meldungs-Handler zur Klasse CHMChart hinzufügen

Ergänzen Sie nun die folgenden Deklarationen gleich zu Beginn von **HMChart.h**:

```
#include "chart.h"

extern "C"
{
    void xxx_chart_paint_hdc(RECT rc, HDC hdc);
    void xxx_chart_set_options(int options);
    void xxx_chart_lbutton(HWND hwnd, int mouse_x, int mouse_y);
    void xxx_chart_rbutton(HWND hwnd, int mouse_x, int mouse_y);
    void xxx_chart_next_frame(HWND hwnd);
}
```

Die folgenden Definitionen gehören in die Datei **HMChart.cpp**:

```
// CHMChart-Meldungshandler
void CHMChart::OnPaint()
{
    CPaintDC dc(this); // device context for painting
    CRect rc;

    GetClientRect(rc);
    HRGN hrgn = CreateRectRgn(rc.left, rc.top, rc.right, rc.bottom);
    SelectClipRgn(dc, hrgn);
    DeleteObject(hrgn);

    HBRUSH white = CreateSolidBrush(RGB(255, 255, 255));
    FillRect(dc, &rc, white);
    DeleteObject(white);
    xxx_chart_paint_hdc(rc, dc);
}

void CHMChart::OnLButtonDown(UINT nFlags, CPoint point)
{
    xxx_chart_lbutton(m_hWnd, point.x, point.y);
}
```

```

void CHMChart::OnRButtonDown(UINT nFlags, CPoint point)
{
    xxx_chart_rbutton(m_hWnd, point.x, point.y);
}

void CHMChart::OnClose()
{
    // CStatic::OnClose();    ...auskommentiert!
}

```

#### 4. Schritt: Chart zur Benutzeroberfläche hinzufügen

Erstellen Sie im Dialogeditor ein neues Element des Typs **Static Text** an derjenigen Position, wo später das Chart erscheinen soll. Als **ID** für das neue Element wird **IDC\_CHART** vergeben, die Eigenschaft **Notify** wird auf **True** gesetzt, siehe Abbildung A3. Dieser letzte Punkt ist wichtig, da sonst das Chart später nicht in der Lage ist, auf Mausklicks zu reagieren.

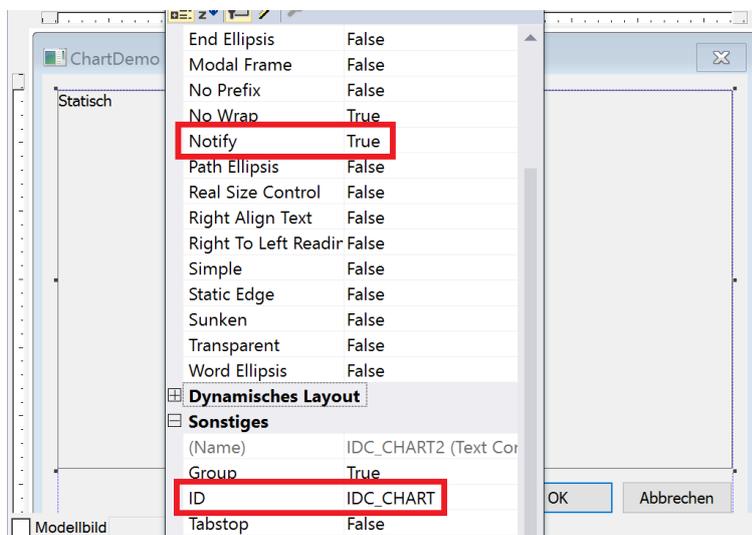


Abbildung A3 – Position des Charts in der Benutzeroberfläche festlegen

Mithilfe des **Assistenten zum Hinzufügen von Membervariablen** wird zum soeben erstellten Steuerelement **IDC\_CHART** eine Membervariable definiert: Der Variablenname lautet **m\_Chart**, als Variablentyp wird **CHMChart** eingetragen, siehe Abbildung A4.

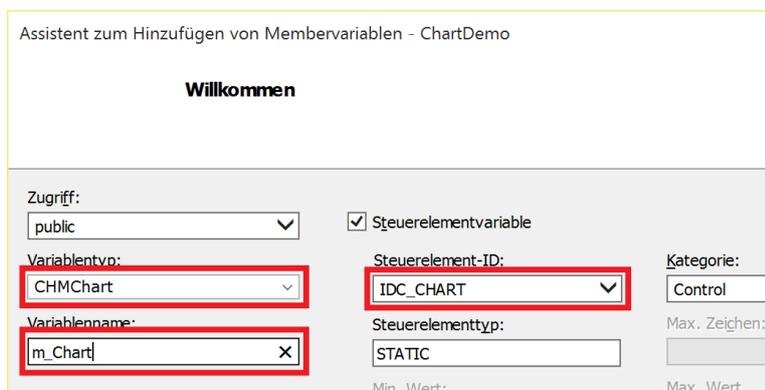


Abbildung A4 – Membervariable für das Chart-Steuerelement definieren

Fügen Sie zu Beginn von **ChartDemoDlg.h** die folgende Include-Anweisung hinzu (direkt hinter der Anweisung **#pragma once**):

```
#include "HMChart.h"
```

Die Daten, die im Chart angezeigt werden sollen, müssen noch an das Chart übergeben werden. Dies kann zum Beispiel in der Methode **CChartDemoDlg::OnInitDialog()** geschehen, welche in der Datei **ChartDemoDlg.cpp** definiert ist:

```
// TODO: Hier zusätzliche Initialisierung einfügen
double x, y;
for(x = -5; x <= 5; x += 0.1)
{
    y = x * x;
    if(x == -5)
        chart_moveto(x, y);
    else
        chart_lineto(x, y, CHART_RED);
}

// Auf Wunsch können auch Chart-Optionen gesetzt werden;
// hier wird zum Beispiel der graue Hintergrund entfernt.
xxx_chart_set_options(CHART_NOBACKGROUND);
return TRUE;
}
```

Schließlich wird die Applikation kompiliert und ausgeführt. Abbildung A5 zeigt die fertige Benutzeroberfläche mit dem integrierten Chart. Mit den beiden Maustasten lässt sich die Ansicht vergrößern, verkleinern oder auch auf einem Drucker ausgeben.

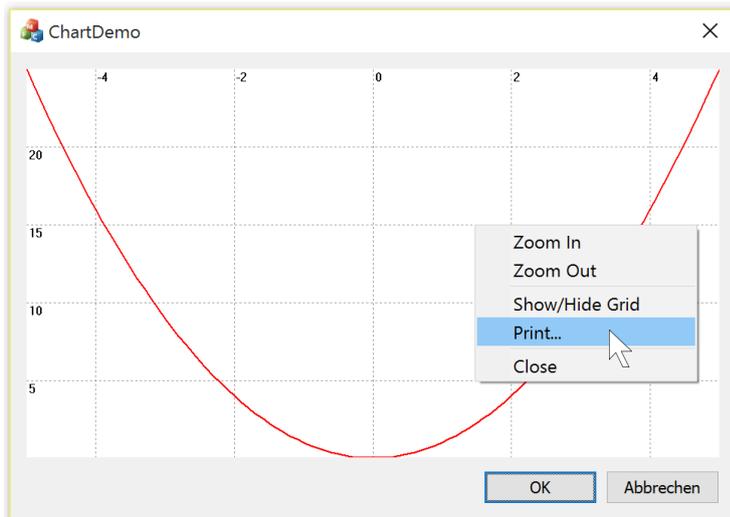


Abbildung A5 – Die fertige Benutzeroberfläche

## 5. Schritt: Animierte Charts

Auch animierte Charts können in bestehende Benutzeroberflächen integriert werden. Zusätzlich zu den bereits genannten Schritten muss dazu noch ein Timer aktiviert werden, welcher die „Weiter-schaltung“ der einzelnen Teilbilder auslöst.

Zunächst wird mit dem Klassenassistenten (Strg+Umschalt+x) ein Handler für **WM\_TIMER**-Ereignisse zum Dialogfenster hinzugefügt, siehe Abbildung A6.

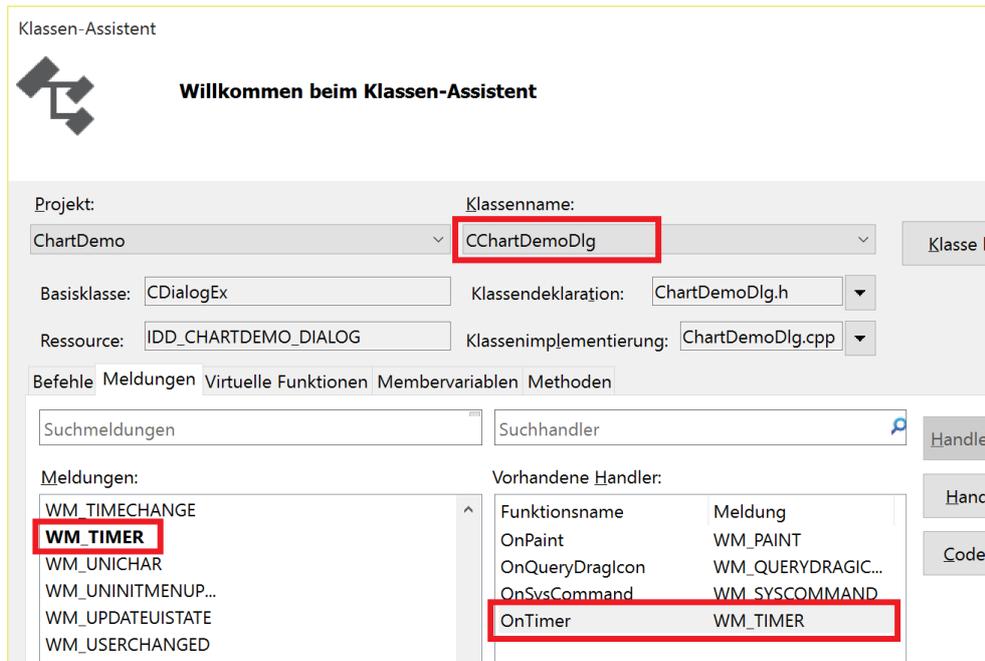


Abbildung A6 – Handler für Timer-Ereignisse hinzufügen

Die Methode **CChartDemoDlg::OnInitDialog()** in der Datei **ChartDemoDlg.cpp** wird geändert:

```
// TODO: Hier zusätzliche Initialisierung einfügen
for(double x1 = 0; x1 <= 5; x1 += 0.1)
    chart_lineto(x1, x1, CHART_RED);

// Erstes Teilbild ist fertig, es folgt das zweite Teilbild...
chart_end_of_frame();

for(double x2 = 0; x2 <= 5; x2 += 0.1)
    chart_lineto(x2, -x2, CHART_BLUE);

// Die Teilbilder werden alle 500 ms umgeschaltet...
SetTimer(0, 500, 0);
return TRUE;
}
```

Und die Implementierung der Methode **CChartDemoDlg::OnTimer()** in derselben Datei lautet:

```
void CChartDemoDlg::OnTimer(UINT_PTR nIDEvent)
{
    xxx_chart_next_frame(m_hWnd);
    // CDialogEx::OnTimer(nIDEvent);    ...auskommentiert!
}
```