9.10 GTK/Zeichenfläche

9.10.1 Graphik-Ausgabe

In vielen Programmen soll eine Zeichnung auf dem Bildschirm ausgegeben werden. Mit den bisherigen Widgets bekommt man jedoch nur Texte, Knöpfe, Schieberegler und andere Standard-Elemente in sein Fenster.

Gebraucht wird eine Sammlung von Funktionen, mit denen man Punkte, Linien und Flächen darstellen kann, also eine 2D-Graphik-Bibliothek; und man braucht die Einbindung dieser Funktionen in GTK.

Bis zur Version 2 von GTK+ benutzt man dazu eine Bibliothek mit dem Namen *GDK* (*gimp drawing toolkit*). Sie liegt technisch recht nahe an der X11-Bibliothek von Unix/Linux.

Seit der Version 3 von GTK+ gibt es zu diesem Zweck eine Bibliothek mit dem Namen *Cairo*. Sie ist eine 2D-Vektor-Bibliothek mit der Möglichkeit, Pixelbilder einzubinden.

Das Zeichenmodell von Cairo sieht wie folgt aus. Die Formen und Texte, die gezeichnet werden sollen, nennt man *Maske*. Die Farbe (oder der Farbverlauf) mit der gezeichnet werden soll, nennt man *Source*. Die Maske drückt die Farbe auf die Zeichenfläche (die nennt man *Surface*). Man kann also die Maske als Stempel und die Source als Stempelkissen verstehen.

Cairo ist vergleichbar mit Quartz (Apple) und GDI (MS), ist aber eine Open-Source-Bibliothek und auf vielen Plattformen verwendbar. Sie wird in zahlreichen Programmen (z. B. Firefox) und Toolkits verwendet und erlaubt eine ganze Reihe von Ausgabeformaten auch außerhalb von GUIs (PDF, PNG u.a.).

9.10.2 Zeichnen mit GTK und Cairo

In `zeichnen0.c` werden zwei Linien gezeichnet.

```
1  #include <gtk/gtk.h>
2  struct gui_t
3  {
4     GtkWidget *fenster;
5     GtkWidget *zeichenflaeche;
6  };
7
8  gboolean zeichnen0(GtkWidget *flaeche, cairo_t *cr, gpointer muell)
9  {
10                                  /* Beginn des Pfades    */
11    cairo_move_to(cr,   0,   0); /* Zu einem Startpunkt */
12    cairo_line_to(cr, 400, 400); /* Linie festlegen     */
13    cairo_move_to(cr,   0, 400); /* Neuer Startpunkt    */
14    cairo_line_to(cr, 400,   0); /* Linie festlegen     */
15    cairo_stroke(cr);            /* Alles zeichnen,     */
16                                  /* Pfad wird dabei     */
17                                  /* geloescht           */
18    return FALSE;
19 }
20 ////////////////////////////////////////////////////////
21 int main(int argc, char *argv[])
22 {
23    struct gui_t gui;
24    gtk_init(&argc, &argv);
25    gui.fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
26    gui.zeichenflaeche=gtk_drawing_area_new();
27    gtk_widget_set_size_request(gui.zeichenflaeche, 300, 300);
28
```

```
29      gtk_container_add(GTK_CONTAINER(gui.fenster), gui.zeichenflaeche);
30
31      g_signal_connect(G_OBJECT(gui.fenster),"delete_event",
32                       gtk_main_quit, NULL);
33      g_signal_connect(G_OBJECT(gui.zeichenflaeche), "draw",
34                       G_CALLBACK(zeichnen0), NULL);
35
36      gtk_widget_show_all(gui.fenster);
37      gtk_main();
38      return 0;
39  }
```

Zeile 26 Ein neues Widget ist die Zeichenfläche. Sie wird hier angelegt. Sie ist zunächst leer und hat noch nicht mal einen Rahmen.

Zeile 27 Hier wird die Größe der Zeichenfläche angelegt (Breite mal Höhe in Pixeln).

Zeile 29 Die Zeichenfläche kann in ein Fenster oder eine Box oder ein sonstiges Widget gepackt werden, wie jedes andere Widget auch.

Zeile 33 Wenn die Zeichenfläche gezeichnet werden soll, wird das Ereignis `"draw"` ausgelöst. Diese Zeile bestimmt, dass in unserem Programm dann die Funktion `zeichnen0` aufgerufen wird.

Bei der älteren Bibliothek GDK heißt das entsprechende Ereignis `"expose-event"`. Das Ereignis wird immer dann ausgelöst, wenn Zeichenfläche (oder ein Teil davon) neu oder wieder sichtbar wird. Das kann zum Beispiel der Fall sein, wenn das Fenster vorher verdeckt oder minimiert war.

Zeile 8 Hier finden wir die zum Ereignis `"draw"` passende Callback-Funktion.

erster Parameter Zeichenfläche aus GTK-Sicht

zweiter Parameter Zeichenfläche aus Cairo-Sicht. `cr` ist als globale Variable bereits vorhanden.

dritter Parameter Zusatzdaten (wie immer).

Die Zeichenfläche wird nur durch diese Callback-Funktion bemalt, nicht durch die Funktion `gtk_widget_show_all()`!

Zeilen 11–15 Hier finden wir die Zeichenfunktionen. Zuerst wird der Pfad festgelegt, das ist die Maske (Zeilen 11–14). Anschließend wird der Pfad auf die Zeichenfläche gedrückt und dabei gelöscht (Zeile 15). Die Farbe (source) ist hier noch standardmäßig schwarz.

Zeile 11 Springen zum Punkt mit der Koordinate x=0, y=0. Das ist der Punkt links oben in der Ecke. Es wird in Pixeln gerechnet. Es wird bis jetzt noch nichts gezeichnet.

Zeile 12 Zeichnen einer Linie vom aktuellen Punkt (0,0) zum neuen Punkt (400,400). Dieser ist jetzt der aktuelle Punkt.

Die Befehle move und line stammen aus der Ansteuerung von Plottern (move: Stift hoch und bewegen, line: Stift runter und bewegen). Da es Vektor-Befehle sind, braucht man sich keine Gedanken über die Punkte zwischen (0,0) und (400,400) zu machen; sie werden automatisch berechnet und dargestellt.

Im nächsten Beispiel-Programm `zeichnen1.c` findet man weitere Möglichkeiten:

```
8  gboolean zeichnen(GtkWidget *flaeche, cairo_t *cr, gpointer muell)
9  {
10     guint breite, hoehe;
11     breite=gtk_widget_get_allocated_width(flaeche);
12     hoehe=gtk_widget_get_allocated_height(flaeche);
13
14     cairo_move_to(cr, 0,      0     );
15     cairo_line_to(cr, breite, hoehe);
16     cairo_move_to(cr, 0,      hoehe);
17     cairo_line_to(cr, breite, 0     );
18     cairo_stroke(cr);
19     return FALSE;
20 }
```

Zeile 11–12 So kommt man in der Callback-Funktion an die aktuelle Größe der Zeichenfläche.

Zeile 15–17 Und so benutzt man sie.

Im folgenden Beispiel-Programm `zeichnen2.c` finden sich weitere Zeichenfunktionen:

```
8  gboolean zeichnen1(GtkWidget *flaeche, cairo_t *cr, gpointer muell)
9  {
10     cairo_rectangle(cr, 0, 0, 50, 50);
11     cairo_stroke(cr); /* geleert durch stroke */
12
13     cairo_rectangle(cr, 100, 100, 50, 50);
14     cairo_fill(cr); /* geleert durch fill */
15
16     cairo_move_to(cr, 200,200);
17     cairo_curve_to(cr,250,300,400,400,400,200);
18     cairo_fill(cr);
19
20     cairo_move_to(cr, 50, 300);
21     cairo_show_text(cr, "Das␣ist␣das␣Haus␣vom␣Nikolaus.");
22     cairo_fill(cr);   /* geleert durch fill */
23
24     return FALSE;
25 }
```

Zeile 10 Rechteck, die numerischen Parameter sind: x-Koordinate links, y-Koordinate oben, Breite und Höhe

Zeile 14 Der bisherige Pfad soll nicht nur gezeichnet, sondern auch ausgefüllt werden.

Zeile 17 Kurve vom aktuellen Punkt entlang der Stützpunkte (x1,y1) und (x2,y2) hin zu (x3,y3). Die numerischen Parameter sind x1, y1, x2, y2, x3 und y3.

Zeile 21 Ein Text wird am aktuellen Punkt angezeigt. Es gibt Funktionen zum Einstellen der Schriftart und der Schriftgröße.

Es gibt in Cairo auch noch eine Funktion, mit der man einen Kreis oder einen Kreisbogen zeichnen kann:

```
1     cairo_arc(cr, x, y, r, phi1, phi2);
```

Die numerischen Parameter sind die Koordinaten des Mittelpunkts, der Radius sowie der Startwinkel und der Endwinkel (beide im Bogenmaß)[1].

9.10.3 Zeichnen mit Farbe

Im Beispiel `zeichnen3.c` sieht man, wie man an Farben herankommt:

```
8  gboolean zeichnen1(GtkWidget *flaeche, cairo_t *cr, gpointer muell)
9  {
10     cairo_set_source_rgba(cr, 0.0, 1.0, 0.0, 0.5); /* Farbe: gruen */
11     cairo_rectangle(cr, 0, 0, 20, 20);
12     cairo_move_to(cr,20,20);
13     cairo_line_to(cr,300,300);
14     cairo_curve_to(cr,400,300,500,400,600,300);
15     cairo_stroke(cr); /* geleert durch stroke */
16
17     cairo_set_source_rgba(cr, 0, 0, 0, 1);          /* Farbe: schwarz */
18     cairo_move_to(cr, 50, 50);
19     cairo_show_text(cr, "Das ist das Haus vom Nikolaus.");
20     cairo_fill(cr);   /* geleert durch fill */
21
22     cairo_set_source_rgba(cr, 1.0, 0.0, 0.0, .5);  /* Farbe: rot */
23     cairo_move_to(cr, 0,200); /* Startpunkt */
24     cairo_line_to(cr, 300,200);
25     cairo_line_to(cr, 300,250);
26     cairo_close_path(cr); /* zum Startpunkt 0,200 */
27     cairo_fill(cr);
28     return FALSE;
29 }
```

Zeile 10 Die Source wird gesetzt. Hier wird das Farbmodell RGBA genommen. Das heißt, es findet additive Farbmischung statt. Die Grundfarben sind rot, grün und blau. Der Alpha-Kanal legt die Transparenz fest. Mit den vier numerischen Parametern wird festgelegt: Helligkeit des Rot-Anteils von `0.0` (dunkel) bis `1.0` (hell), Helligkeit des Grün-Anteils, Helligkeit des Blau-Anteils, Transparenz der Farbe von `0.0` (durchsichtig) bis `1.0` (komplett überdeckend, undurchsichtig)

Man kann sich einen Farbwert auch vom GTK-/GDK-Standard-Hintergrund holen. Ein Beispiel sieht man in `zeichnen4.c` Außerdem sind auch Farbverläufe möglich.

[1] Man kann auch elliptische Bögen zeichnen; dazu gibt es die Möglichkeit der Koordinaten-Transformation.