9.12 GTK/Timer

9.12.1 Ungelöste Probleme

Mit den bis jetzt vorgestellten Widgets und Ereignissen kann man schon sehr viele GUI-Programme schreiben. Es bleiben aber noch einige Wünsche offen.

- Ein Spielprogramm ist zu schreiben: Das Programm zeichnet alle 100 Millisekunden an zufälliger Stelle einen Kreis auf die Zeichenfläche. Der Benutzer muss so schnell wie möglich den Kreis anklicken.
- Timeout einer Benutzereingabe: Das Vokabelprogramm wartet nur eine bestimmte Zeit auf das Ergebnis. Danach wird die Eingabe ausgewertet.

9.12.2 Die Timer-Funktion

In der Funktionsbibliothek GLib, die von GTK+ benutzt wird, findet man eine Funktion für einen Timer. Sie heißt `g_timeout_add()` und bewirkt, dass eine bestimmte Callback-Funktion aufgerufen wird. Anders als bei Knöpfen oder Checkboxen braucht man dafür als Anwender nicht zu klicken. Stattdessen wird die Funktion nach einer bestimmten Zeit von selbst aufgerufen. Voraussetzung dafür ist, dass das Programm sich in dieser Zeit in der Hauptschleife des Programms befindet. Aber das ist ja bei den bisherigen Ereignissen genauso.

```
1    guint timer_nr;
2    timer_nr=g_timeout_add(1200, cbfunktion, daten);
```

Genau 1200 ms nach dieser Zeile wird die Funktion `cbfunktion()` aufgerufen. Sie hat folgenden Prototyp:

```
1 gboolean funktion(gpointer daten);
```

Anders als bei den Callback-Funktionen für gedrückte Knöpfe und Ähnliches ist hier der Rückgabewert entscheidend:

- `FALSE`: Der Timer wird abgeschaltet. Die Funktion wird durch *diesen* Timer nie wieder aufgerufen. Beispiel: `src/timer00.c`
- `TRUE`: Der Timer bleibt eingeschaltet. Nach weiteren 1200 ms wird diese Funktion schon wieder aufgerufen. Beispiele: `src/timer01.c` und `src/timer02.c` (mit Zähler)

Mit `FALSE` als Rückgabewert kann man also einen einmaligen Aufruf realisieren. Mit `TRUE` als Rückgabewert kann man einen wiederholten Aufruf verwirklichen. Man kann auch zehnmal `TRUE` zurückgeben und einmal `FALSE`. Dann hat man elf Aufrufe im Abstand von jeweils 1200 ms.

Ein Beispiel sieht man in `src/timer03.c` [1]:

```
1  #include <gtk/gtk.h>
2  gboolean fertig(gpointer daten)
3  {
4      static int anzahl=0;
5      ++anzahl;
6      g_print("%i mal aufgerufen\n", anzahl);
7      if(anzahl==11)
8          return FALSE;
9      else
10         return TRUE;
11 }
```

[1] In den Beispielen `src/timer10.c` bis `src/timer16.c` findet man die gleichen Programme mit graphischer Ausgabe.

```
12 int main(int argc, char *argv[])
13 {
14     GtkWidget *fenster;
15     gtk_init(&argc, &argv);
16     fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
17     g_signal_connect(G_OBJECT(fenster),"delete-event",
18                      gtk_main_quit, NULL);
19     g_timeout_add(1000, fertig, NULL);
20
21     gtk_widget_show_all(fenster);
22     gtk_main();
23     return 0;
24 }
```

Wenn man möchte, dass die Callback-Funktion schon beim Programmstart sofort aufgerufen wird, dann muss man einen Aufruf von `cbfunktion()` direkt nach dem Aufruf von `g_timeout_add()` einfügen. So ist es in `src/timer04.c`:

```
20     fertig(NULL);
```

9.12.3 Wechselnde Taktzeiten

Man kann die Funktion `g_timeout_add()` auch in der Callback-Funktion selbst aufrufen. Auf diese Art kann man die Taktzeiten variieren. Man muss aber aufpassen (`src/timer05.c`):

```
2  gboolean fertig(gpointer daten)
3  {
4      static int anzahl=0;
5      ++anzahl;
6      g_print("%i␣mal␣aufgerufen\n", anzahl);
7      if(anzahl%2!=0) /* ungerade */
8          g_timeout_add(200, fertig, NULL);
9      else
10         g_timeout_add(2000, fertig, NULL);
11     return TRUE;
12 }
```

Wenn man es so macht, steigt die Anzahl der Aufrufe plötzlich sehr stark. Das liegt daran, dass durch die erneuten Aufrufe von `g_timeout_add()` mehrere Timer aktiviert wurden, die nun alle gleichzeitig laufen und wiederum die Callback-Funktion aufrufen.

Damit das nicht passiert, damit also immer nur ein Timer gleichzeitig läuft, muss man immer dann, wenn ein neuer Timer gestartet wird, die Callback-Funktion mit `FALSE` verlassen (`src/timer06.c`):

```
11     return FALSE;
```

9.12.4 Abschalten eines Timers

Will man einen Timer asynchron (z. B. vor dem ersten Aufruf) stoppen, ruft man auf:

```
1      g_source_remove(timer_nr);
```

Dazu muss man sich allerdings den Rückgabewert von `g_timeout_add()` merken. Er dient als Parameter für die Funktion `g_source_remove()`.