## 9.2 GTK/Callback-Funktionen

### 9.2.1 Beendigung eines GTK-Programms

Das erste Programm konnte noch nicht auf graphische Weise, etwa durch Anklicken des X-Symbols in der Titelleiste, beendet werden. Das soll nun anders werden. `beispiel2.c` zeigt, wie es geht:

```
1  #include <gtk/gtk.h>
2  int main(int argc, char *argv[])
3  {
4      GtkWidget* fenster;
5
6      gtk_init(&argc, &argv);
7      fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
8      g_signal_connect(G_OBJECT(fenster),"delete_event",gtk_main_quit,
9                       NULL);
10     gtk_widget_show_all(fenster);
11     gtk_main();
12     return 0;
13 }
```

### 9.2.2 Bedeutung der Programmzeilen

**Zeile 8** Hier wird ein Funktionszeiger, nämlich die Adresse der Beendigungsfunktion `gtk_main_quit()`, in einen Platz der Datenstruktur des Hauptfensters geschrieben. Immer, wenn das Hauptfenster ab jetzt ein Ereignis mit dem Namen `delete_event` erhält, wird diese Funktion durch GTK aufgerufen und beendet das Programm. Die Funktion wird hier also als Callback-Funktion benutzt. Man sagt, das Ereignis `delete_event` wird für das Hauptfenster mit der Funktion `gtk_main_quit()` *verbunden*.

Diese Funktion gibt es übrigens schon, so dass wir sie nicht selbst schreiben müssen.

**Zeile 11** In der Hauptschleife wird jetzt selbständig die Funktion `gtk_main_quit()` aufgerufen, sobald das Ereignis `delete_event` aufgetreten ist.

Eine Funktion wie `gtk_main_quit()` kann man sich auch selbst schreiben. Man bekommt dann die Möglichkeit, selbst auf bestimmte Ereignisse zu reagieren.

### 9.2.3 Callback-Funktion selbst schreiben

Wie kann man für Ereignisse (Nutzer klickt Feld usw.) eigene Aktionen bekommen? Man schreibt sich einfach eine eigene Callback-Funktion und trägt sie dann ein (`beispiel3.c`):

```
1  #include <gtk/gtk.h>
2  gboolean endefunk(GtkWidget *fenster,GdkEvent ereignis,gpointer muell)
3  {
4      g_print("Das␣Fenster␣wird␣entfernt.\n");
5      return TRUE;
6  }
7
8  int main(int argc, char *argv[])
9  {
10     GtkWidget* fenster;
11
12     gtk_init(&argc, &argv);
13     fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
```

```
14      g_signal_connect(G_OBJECT(fenster), "delete_event",
15                       G_CALLBACK(endefunk), NULL);
16      gtk_widget_show_all(fenster);
17      gtk_main();
18      return 0;
19  }
```

**Zeile 14** Hier wird für das Ereignis `delete_event` die selbstgeschriebene Funktion `endefunk()` als Callback-Funktion eingesetzt[1].

**Zeile 2** Hier beginnt die Callback-Funktion. Erster Parameter ist immer das Widget, welches das Ereignis erhalten hat. Zweiter Parameter ist die Art des Ereignisses. Als letzter Parameter wird ein Zeiger auf irgendwelche zusätzliche Daten angegeben, meistens braucht man ihn nicht.

**Zeile 4** Ja, auch GUI-Programme dürfen auf die Konsole schreiben.

**Zeile 5** Bei Rückgabe von `FALSE` (also 0) wird die Ereignisbearbeitung an GTK zurückgegeben, und es kann die nächste Callback-Funktion ausgeführt werden. Falls keine weitere Callback-Funktion vorhanden ist, wird die normale GTK-Ereignisbehandlung ausgeführt. Beim Ereignis `"delete-event"` bedeutet das, es wird dann ein destroy-*Signal* ausgesandt[2]. Dadurch wird das Fenster zerstört. Das Programm wird jedoch nicht beendet.

Bei Rückgabe von `TRUE` (also 1) gilt die Ereignisbearbeitung für GTK als beendet. Es wird nichts weiter getan. Das Widget bleibt erhalten.

### 9.2.4 Mehrere Callback-Funktionen

In `beispiel4.c` sind nun mehrere Callback-Funktionen eingerichtet, die mit verschiedenen Ereignissen verbunden sind:

```
1   #include <gtk/gtk.h>
2   gboolean endefunk(GtkWidget *fenster,GdkEvent ereignis,gpointer m)
3   {
4       g_print("Das␣Fenster␣wird␣entfernt.\n");
5       gtk_main_quit(); /* Programm-Ende      */
6       return TRUE;     /* wird nie erreicht */
7   }
8   gboolean focusinfunk(GtkWidget *fenster,GdkEvent ereignis,gpointer m)
9   {
10      g_print("Bin␣aktiv.\n");
11      return TRUE;
12  }
13  gboolean focusoutfunk(GtkWidget *fenster,GdkEvent ereignis,gpointer m)
14  {
15      g_print("Bin␣passiv.␣");
16      return FALSE;
17  }
18  gboolean focusoutfunk2(GtkWidget *fenster,GdkEvent ereignis,gpointer m)
19  {
20      g_print("Wirklich␣passiv\n");
21      return TRUE;
```

[1] In GTK 1.2 hieß der Funktionsname `gtk_signal_connect()`; er soll nicht mehr benutzt werden.

[2] Ein Signal ist vergleichbar mit einem Ereignis; nur wird ein Signal vom Programm selbst erzeugt. Auch ein Signal kann per `g_signal_connect()` mit einer Callback-Funktion verbunden werden. Diese Callback-Funktionen für Signale sind zum Teil Widget-spezifisch. Man erkennt sie am Rückgabetyp `void`.

```
22 }
23 int main(int argc, char *argv[])
24 {
25     GtkWidget* fenster;
26
27     gtk_init(&argc, &argv);
28     fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
29     g_signal_connect(G_OBJECT(fenster),"delete-event",
30                      G_CALLBACK(endefunk),NULL);
31     g_signal_connect(G_OBJECT(fenster),"focus-in-event",
32                      G_CALLBACK(focusinfunk),NULL);
33     g_signal_connect(G_OBJECT(fenster),"focus-out-event",
34                      G_CALLBACK(focusoutfunk),NULL);
35     g_signal_connect(G_OBJECT(fenster),"focus-out-event",
36                      G_CALLBACK(focusoutfunk2),NULL);
37     gtk_widget_show_all(fenster);
38     gtk_main();
39     return 0;
40 }
```

**Zeilen 29 bis 36** `delete-event` und `focus-in-event` sind mit je einer Callback-Funktion verbunden, `focus-out-event` ist gleich mit zwei Funktionen verbunden.

**Zeile 5** Beim Entfernen des Fensters wird das Programm durch den Aufruf von `gtk_main_quit()` beendet.

**Zeile 6** Diese Zeile wird nicht erreicht und steht nur da, um den Compiler nicht zu irritieren.

**Zeile 16** Die Funktion reicht durch `FALSE` das Ereignis weiter an das GTK, das die nächste Callback-Funktion aufruft.

**Zeile 21** Die Funktion erklärt die Ereignisbehandlung durch `TRUE` für beendet.