9.11 GTK/Mausklicks

9.11.1 Wozu mit Mausklicks befassen?

Der Benutzer bekommt eine Zeichenfläche mit einem Liniendiagramm angezeigt. Wenn er jetzt auf einen Punkt des Diagramms klickt, sollen ihm der x- und der y-Wert des Punktes angezeigt werden, so dass er genau nachsehen kann, an welchen x- und y-Werten des Diagramms interessante Stellen sind.

9.11.2 Erkennen von Mausklicks

Wenn mit der Maus auf der Zeichenfläche geklickt wird, sollen in der Regel einige Dinge gemeldet werden:

- x- und y-Koordinate
- Nummer der Maustaste,
- Wurde die Taste gedrückt (ein-, zwei- oder dreimal) oder nur losgelassen?

In GTK reagiert man auf den Mausklick mit dem Auruf einer Callback-Funktion. Wie das geht, zeigt `klick1.c`.

```c
#include <gtk/gtk.h>
struct gui_t
{
   GtkWidget *fenster;
   GtkWidget *zeichenflaeche;
};
gboolean klick(GtkWidget *flaeche, GdkEvent *ereignis, gpointer daten)
{
   char *str=daten;
   switch(str[0])
   {
      case 'p': g_print("Maustaste gedrueckt\n");    break;
      case 'r': g_print("Maustaste losgelassen\n"); break;
      default:  g_print("- unbekannt -\n");          break;
   }
   return FALSE;
}
gboolean zeichnen(GtkWidget *flaeche, cairo_t *cr, gpointer muell)
{
   cairo_arc(cr, 100, 100, 80, 0, 2.0*G_PI);
   cairo_stroke(cr);
   return FALSE;
}
int main(int argc, char *argv[])
{
   struct gui_t gui;
   gtk_init(&argc, &argv);
   gui.fenster=gtk_window_new(GTK_WINDOW_TOPLEVEL);
   gui.zeichenflaeche=gtk_drawing_area_new();
   gtk_widget_set_size_request(gui.zeichenflaeche, 200, 200);
   gtk_widget_add_events(gui.zeichenflaeche, GDK_BUTTON_PRESS_MASK);
   gtk_widget_add_events(gui.zeichenflaeche, GDK_BUTTON_RELEASE_MASK);

   gtk_container_add(GTK_CONTAINER(gui.fenster), gui.zeichenflaeche);
```

```
35
36      g_signal_connect(G_OBJECT(gui.fenster),"delete_event",
37                       gtk_main_quit, NULL);
38      g_signal_connect(G_OBJECT(gui.zeichenflaeche), "draw",
39                       G_CALLBACK(zeichnen), NULL);
40      g_signal_connect(G_OBJECT(gui.zeichenflaeche),
41                       "button_press_event",  G_CALLBACK(klick), "p");
42      g_signal_connect(G_OBJECT(gui.zeichenflaeche),
43                       "button_release_event", G_CALLBACK(klick), "r");
44      gtk_widget_show_all(gui.fenster);
45      gtk_main();
46      return 0;
47  }
```

Zeile 31 Wenn sich der Mauscursor auf der Zeichenfläche befindet, soll beim Drücken einer Maustaste ein Ereignis ausgelöst werden. Das wird mit dieser Zeile festgelegt.

Zeile 32 Auch beim Loslassen einer Maustaste soll ein Ereignis ausgelöst werden. Das `add` bezieht sich darauf, dass man nacheinander weitere Ereignisse hinzufügen kann. Für berührungssensitive Bildschirme gibt es zum Beispiel ein eigenes Ereignis.

Das Kürzel GDK deutet darauf hin, dass die Ereignisse von einer Hardware-näheren Schicht (GDK) stammen.

Zeilen 40-41 Jetzt müssen die Ereignisse mit den passenden Callback-Funktionen verbunden werden.

Es wird die gleiche Funktion genommen (`klick()`); durch den letzten Parameter wird unterschieden, ob Drücken oder Loslassen vorliegt.

Zeile 7-17 Hier ist die Callback-Funktion. Wieder gilt:

a) 1. Parameter = Widget,

b) 2. Parameter = Ereignis,

c) 3. Parameter = Sonstiges

9.11.3 Holen der Ereignis-Daten

Nun soll nicht nur gemeldet werden, dass eine Maustaste gedrückt oder losgelassen wurde. Es sollen auch die entsprechenden Umstände genannt werden. Wie das geht, zeigt `klick2a.c`.

```
7   gboolean klick(GtkWidget *flaeche, GdkEvent *ereignis, gpointer daten)
8   {
9       gdouble x,y;
10      guint   maustastenr, status, typ;
11      guint32 zeit; /* uptime in Millisekunden */
12      char *str=daten;
13      GdkEventButton *mausklick=ereignis; /* umtopfen in richtigen event */
14
15      x=mausklick->x;
16      y=mausklick->y;
17      maustastenr=mausklick->button;
18      status=mausklick->state;
19      typ=mausklick->type;
20      zeit=mausklick->time;
21      g_print("Ereignis: %c Maustaste Nr. %u an x=%f y=%f, "
```

```
22              "Status:%u, Typ:%u, Zeit:%lu\n", str[0],
23               maustastenr, x, y, status, typ, zeit);
24     return FALSE;
25 }
```

Zeile 13 Anpassen des Datentyps. Hier wird das Ereignis (`GtkEvent`) der Bequemlichkeit halber in einen anderen Datentyp gepackt (`GtkEventButton`), der die benötigten Daten leichter herausgibt.

Zeilen 15-20 Die wichtigsten Elemente des Ereignisses werden in Variablen kopiert.

a) `type`: Typ des Ereignisses, eine Nummer. Es gibt viele Ereignisse in einem System wie z. B. Tastendrücke, Schließen eines Fensters (bekannt als `"delete-event"`) und natürlich Mausklicks. Hier interessieren von den z. Zt. 41 Ereignissen nur die vier, die sich auf Mausklicks beziehen (Tabelle 1).

Ereignis Nr.	Bedeutung
4	Maustaste gedrückt (einfacher Klick)
5	Maustaste gedrückt (Doppelklick)
6	Maustaste gedrückt (Dreifachklick)
7	Maustaste losgelassen

Tabelle 1: Bedeutung des Werts in `GdkEventButton.type`

b) `button`: Nummer der Maustaste (eins bis drei), die das Ereignis ausgelöst hat[1].

c) `x, y`: Koordinaten als Gleitkommazahlen

d) `time`: Uhrzeit des Ereignisses, gerechnet in Millisekunden, bezogen auf die *Uptime* (Betriebszeit) des Systems.

e) `state`: Falls der `type` des Ereignisses nicht ausreicht, kann man hier genauer ermitteln, was passiert ist. Man erhält eine Bitmaske (Tabelle 2)

Bit Nr.	Bedeutung
0	Umschalttaste an
2	Strg-Taste an
4	NumLock an
6	Rechte Win-Taste an
7	Linke Win-Taste an
8	Maustaste 1 wurde losgelassen
9	Maustaste 2 wurde losgelassen
10	Maustaste 3 wurde losgelassen

Tabelle 2: Bedeutung der Bits in `GdkEventButton.state`

[1] Für die Maustasten ab Nummer vier, die durch das Scrollrad der Maus ausgelöst werden, gibt es eigene Ereignisse.