## 9.9 GTK/Glade und GTK-Builder

### 9.9.1 Was ist Glade?

Glade ist ein Programm, mit dem man die graphische Oberfläche eines GTK-Programms ohne Schreiben irgendwelcher Programmzeilen zusammenbauen kann (Abbildung 1).

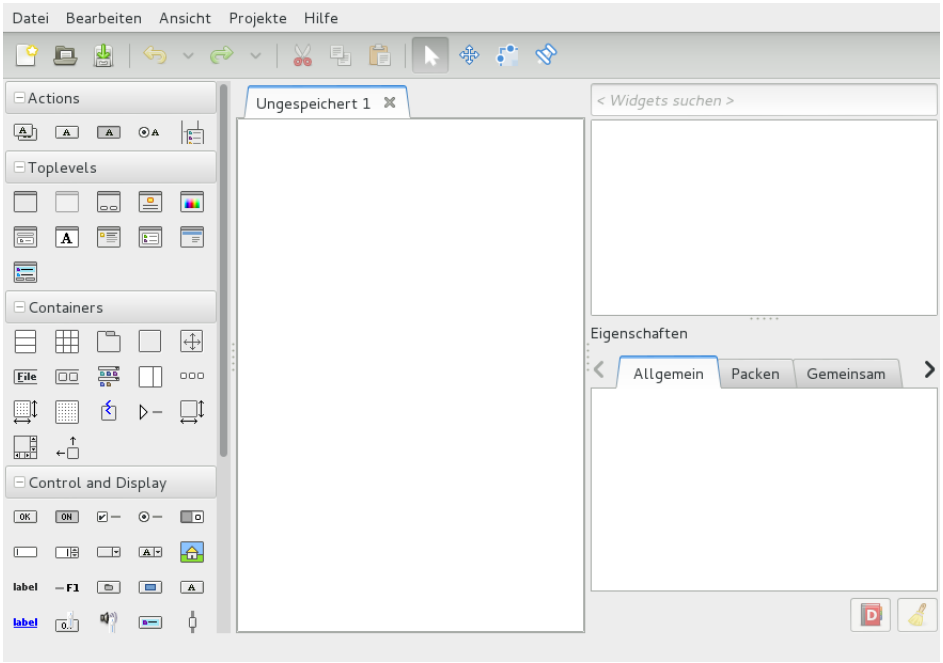


Abbildung 1: Glade

Die Callback-Funktionen und ein (kleines) Hauptprogramm muss man weiterhin in seiner gewohnten Sprache schreiben. Insofern unterscheidet sich Glade von Delphi bzw. Lazarus und auch von Visual Basic.

### 9.9.2 Erstellen einer GUI mit Glade

Nach dem Öffnen von Glade baut man zuerst ein Fenster. Dazu klickt man in der Werkzeugleiste (linkes Teilfenster) im Abschnitt Toplevel auf das Symbol eines Fensters (Abbildung 2). Im mittleren Teilfenster erscheint eine graue Fläche, die das zukünftige Fenster des Programms symbolisieren soll.

Nun soll im Fenster ein Kasten angelegt werden. Dazu klickt man im Abschnitt Containers auf das Symbol Box und anschließend auf die graue Fläche des Programmfensters. Dem sofort aufklappenden Dialog teilt man mit, dass der Kasten zwei Fächer haben soll (Abbildung 3).

Nun sollen ein Knopf und ein Etikett angelegt werden. Dazu gibt es in der Werkzeugleiste die Symbole Button und Label. Der Knopf wird in das untere Fach des Kastens gelegt, das Etikett in das obere (Abbildung 4).

Im Teilfenster links oben sieht man einen Baum aus Elementen. Man kann nun jedes der einzelnen Element anklicken. Eventuell muss man dafür auf das Pluszeichen vor dem Namen eines Elements klicken, damit ein weiterer Teil des Baumes sichtbar wird.

Klickt man nun auf ein Element im Baum, dann kann man in dem Teilfenster darunter (also rechts unten) die Darstellung aller möglichen Eigenschaften des Elements sehen und verändern.

So kann z. B. das Fenster den Namen `fenster` bekommen (im Karteikartenreiter *Allgemein*). Im C-Programm wird das die ID (Identifikation), unter dem man auf das zughörige Widget wird zugreifen können. Ebenso kann man auch den Fenstertitel und andere Eigenschaften ändern, z. B. im

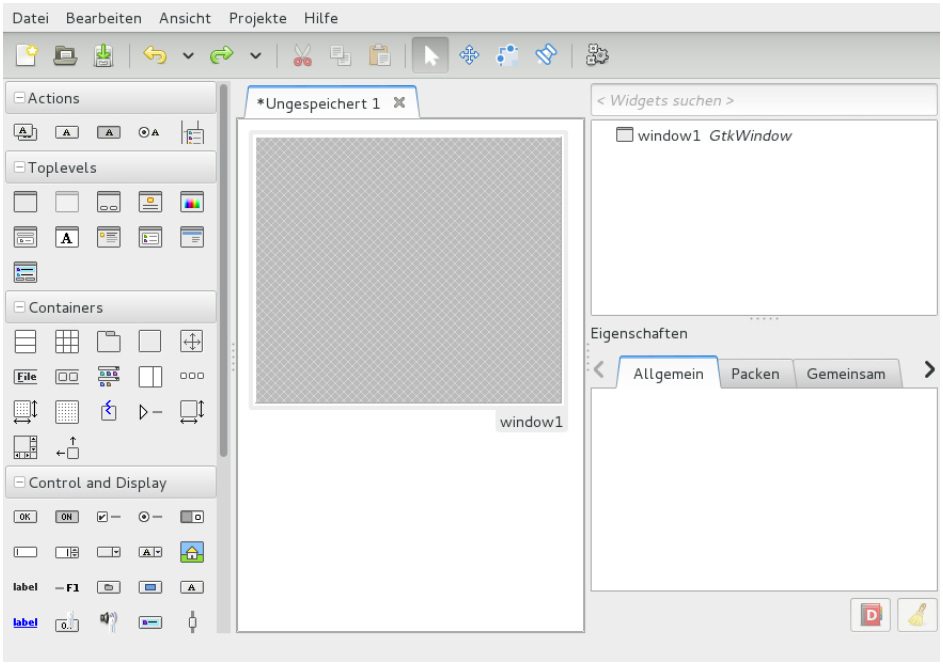


Abbildung 2: Fenster

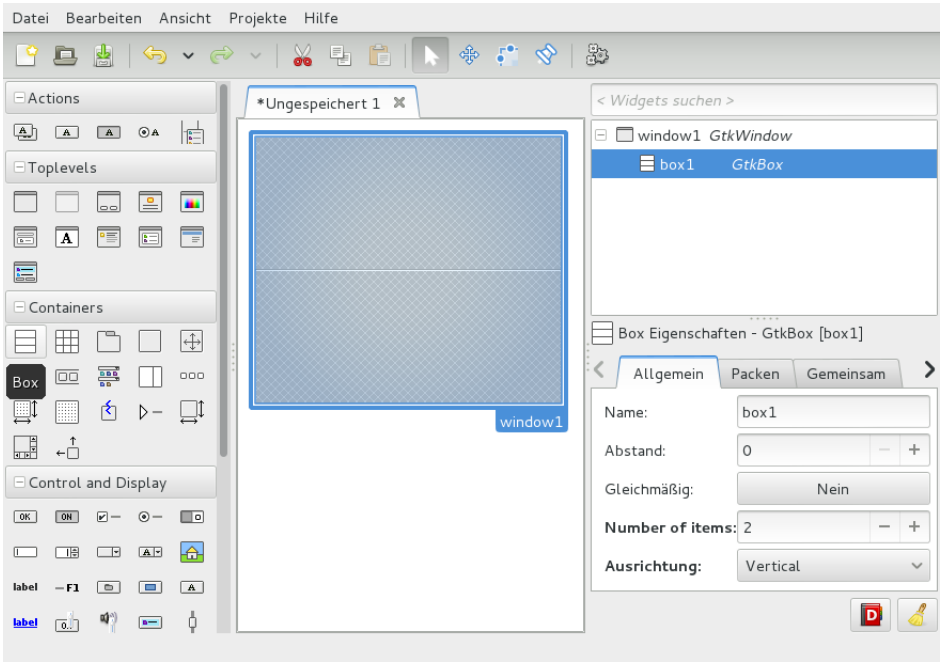


Abbildung 3: Kasten

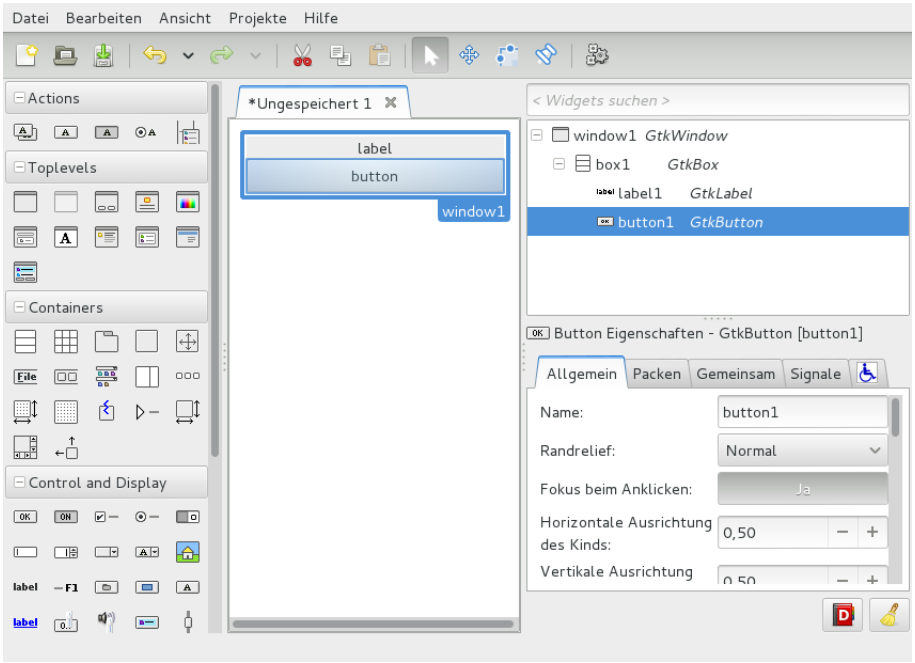


Abbildung 4: Knopf und Etikett

Karteikartenreiter *Gemeinsam* die Mindestbreite (auf 300 Pixel) und -höhe (auf 50 Pixel). Beim Knopf und beim Etikett ist es sinnvoll, wiederum den Namen zu ändern (`knopf` und `etikett`) und die Beschriftung (`Hier klicken` und `Beispiel`).

Wenn man auf das Symbol mit den Zahnrädern in der oberen Werkzeugleiste (unterhalb der Menüleiste) klickt, bekommt man eine Vorschau zu sehen: So (ungefähr) wird unser Programm später auch aussehen.

Nun kann man diese Oberfläche speichern, und zwar am besten im gleichen Verzeichnis wie das Programm (Name: `builder1.ui`). Schaut man sich nach dem Verlassen von Glade die entstandene Datei mit einem Editor an, so sieht man, dass es sich um eine XML-Datei handelt:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <interface>
3    <!-- interface-requires gtk+ 3.0 -->
4    <object class="GtkWindow" id="fenster">
5      <property name="can_focus">False</property>
6      <property name="title" translatable="yes">Beispiel-Programm</property>
7      <property name="default_width">300</property>
8      <property name="default_height">50</property>
9      <child>
10       <object class="GtkBox" id="box1">
11         <property name="visible">True</property>
12         <property name="can_focus">False</property>
13         <property name="orientation">vertical</property>
14         <child>
15           <object class="GtkLabel" id="etikett">
16             <property name="visible">True</property>
17             <property name="can_focus">False</property>
18             <property name="label" translatable="yes">Beispiel</property>
19           </object>
20           <packing>
21             <property name="expand">False</property>
22             <property name="fill">True</property>
23             <property name="position">0</property>
24           </packing>
25         </child>
26         <child>
```

```
27            <object class="GtkButton" id="knopf">
28              <property name="label" translatable="yes">Hier klicken</property>
29              <property name="use_action_appearance">False</property>
30              <property name="visible">True</property>
31              <property name="can_focus">True</property>
32              <property name="receives_default">True</property>
33              <property name="use_action_appearance">False</property>
34              <property name="yalign">0.52999997138977051</property>
35              <property name="image_position">right</property>
36            </object>
37            <packing>
38              <property name="expand">False</property>
39              <property name="fill">True</property>
40              <property name="position">1</property>
41            </packing>
42          </child>
43        </object>
44      </child>
45    </object>
46  </interface>
```

Tatsächlich kann man die Oberfläche für ein Programm auch so aufbauen, dass man die XML-Datei von Hand aufbaut. Manche machen es tatsächlich so.

### 9.9.3 Nutzen der mit Glade aufgebauten GUI

Jetzt brauchen wir nur noch ein C-Programm, das die frisch aufgebaute XML-Datei benutzt, um ihre GUI darzustellen. Wie das geht, sieht man hier (`builder1.c`):

```c
#include <gtk/gtk.h>
void knopfgeklickt(GtkWidget *knopf, gpointer muell)
{
  g_print("Hallo!\n");
}

int main(int argc, char *argv[])
{
  GtkBuilder *baum;
  GObject *element;

  gtk_init(&argc, &argv);

  baum=gtk_builder_new();
  gtk_builder_add_from_file(baum, "builder1.ui", NULL);

  element=gtk_builder_get_object(baum, "fenster");
  g_signal_connect(element, "delete_event",
                   gtk_main_quit, NULL);

  element=gtk_builder_get_object(baum, "knopf");
  g_signal_connect(element, "clicked",
                   G_CALLBACK(knopfgeklickt), NULL);

  element=gtk_builder_get_object(baum, "fenster");
  gtk_widget_show_all(GTK_WIDGET(element));
  gtk_main();
  return 0;
}
```

**Zeile 9:** Dies ist die Variable, die quasi einen Zeiger auf die Elemente der GUI aufnimmt.

**Zeile 10:** Hier ist ein Zeiger auf ein Widget. Der Datentyp ist etwas allgemeiner gehalten als `GtkWidget`. Hier wird `GObject` verwendet, weil das der Rückgabetyp einer häufig benutzten Funktion ist (Zeilen 17 und 25).

**Zeile 14:** Die Datenstruktur für die GUI wird angelegt.

**Zeile 15:** Die XML-Datei wird eingelesen. Die in der XML-Datei beschriebenen Elemente werden angelegt (`gtk_label_new()` usw.). Sie werden ineinander verschachtelt (Kasten in Fenster, Knopf und Etikett in Kasten). Und sie werden in die Datenstruktur aus Zeile 14 eingehängt, so dass man später einfach an sie herankommt.

Erster Parameter ist die Datenstruktur, in die eingehängt werden soll, zweiter Parameter der Name der XML-Datei (relativer oder absoluter Pfadname). Mit dem dritten Parameter hat man die Möglichkeit, sich einen Fehlercode geben zu lassen, das wird hier nicht benutzt[1].

**Zeile 17:** Der Ernstfall tritt ein: Wenn jemand auf das Schließen-Symbol des Fensters klickt, bekommt das Fenster das Ereignis `"delete-event"`. Dann soll die bekannte Funktion `gtk_main_quit()` aufgerufen werden. Dafür gibt es die Funktion `g_signal_connect()`. Und die wiederum braucht einen Zeiger auf das Widget. Den Zeiger bekommt man mit der Funktion `gtk_builder_get_object()`.

Als ersten Parameter gibt man die Datenstruktur an, in die das Widget eingehängt wurde. Zweiter Parameter ist der Name, den man in Glade für das Element gewählt hatte (in der XML-Datei als Attribut `"id"` bezeichnet).

**Zeile 21:** Auch hier braucht man einen Zeiger auf ein Widget. Es wird die gleiche Funktion wie in Zeile 17 verwendet, die gleiche Variable für die Rückgabe, nur der zweite Parameter `"knopf"` sorgt dafür, dass ein anderes Element ausgewählt wird.

**Zeile 24:** Und hier wird zum dritten Mal der Zeiger auf ein Widget geholt, diesmal wieder das `"fenster"`.

**Zeile 25:** Das Anzeigen der GUI wird wie immer bestellt. Da die Variable `element` vom Typ `GObject` ist, sollte man hier (damit der Compiler nicht warnt) das Makro `GTK_WIDGET()` nutzen.

[1] Man kann hier die Adresse eines Zeigers auf ein Record übergeben; dieses Record enthält dann u. a. eine für Menschen lesbare Fehlermeldung.