GUI–Programmierung mit GTK – Teil 3

by Özcan Güngör
<ozcangungor(at)netscape.net>

About the author:

Zur Zeit leiste ich meinen Wehrdienst als Linux–, Oracle–Administrator und Web–Programmierer.

Abstract:

In dieser Serie werden wir lernen, wie man graphische Benutzeroberflächen (GUIs) mit GTK schreibt. Ich habe keine Ahnung, wie viele Folgen sie umfassen wird. Um diese Artikel zu verstehen, solltest du über folgendes in der Programmiersprache C Bescheid wissen:

- Variablen
- Funktionen
- Zeiger

Es empfiehlt sich, die vorherigen Artikel zu lesen:
GUI–Programmierung mit GTK und
GUI–Programmierung mit GTK – Teil 2.
Dieser Artikel ist ein bißchen kürzer als die anderen, weil ich gerade meinen Wehrdienst leisten muß.

Der Toggle–Button

Dieser Button sieht ganz normal aus, aber er hat zwei Zustände: gedrückt oder nicht. Um einen Toggle–Button zu erzeugen, benutzt man eine der folgenden Funktionen:

```
GtkWidget *toggle=gtk_toggle_button_new(void);
GtkWidget *toggle=gtk_toggle_button_new_with_label(const gchar *label);
```

Die erste Funktion erzeugt einen Toggle–Button ohne Text, aber die zweite erzeugt einen, auf dem *label* steht. Du kannst seinen Zustand mit folgender Funktion festlegen:

```
gtk_toggle_button_set_active (GtkToggleButton *toggle_button,
                              gboolean is_active);
```

Dabei ist *toggle_button* der Button, dessen Zustand du ändern willst, und *is_active* ist der Zustand (0 oder 1). Wenn er 0 ist, kann der Button nicht gedrückt werden, ansonsten schon.

Um den Zustand des Buttons zu erfahren, kann folgende Funktion benutzt werden:

```
gboolean gtk_toggle_button_get_active(GtkToggleButton *button);
```

Das Ereignis *toggled* kann mit einem Toggle−Button verbunden werden.

Hier ist ein Beispiel:

```
#include <gtk/gtk.h>
void togg(GtkWidget *widget, gpointer *data){
   if (gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(data)))
    g_print("State is 1\n");
   else
    g_print("State is 0\n");
}

int main( int argc,char *argv[] )
   {

   GtkWidget *window;
   GtkWidget *button;

   gtk_init (&argc, &argv);

   /* Create a new window */
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title (GTK_WINDOW (window), "Toggle Button");

   /* Connect destroy event to the window. */
   gtk_signal_connect (GTK_OBJECT (window), "destroy",
                       GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

   /* Creates a toggle button */
   button=gtk_toggle_button_new_with_label("I'm a toggle button");

   /* Add the button to window */
   gtk_container_add(GTK_CONTAINER(window),button);

   /* Connect "toggled" event to the button */
   gtk_signal_connect (GTK_OBJECT (button), "toggled",
                       GTK_SIGNAL_FUNC(togg),(gpointer *)button);

   gtk_widget_show(button);
   gtk_widget_show (window);

   gtk_main ();
   return(0);
}
```

Der Check−Button

Der Check−Button, der auch Checkbox genannt wird, ist eine Spezialisierung des Toggle−Buttons. Er kann benutzt werden, um Optionen zu wählen.
Ein Check−Button kann mit folgenden Funktionen erzeugt werden:

```
GtkWidget* gtk_check_button_new (void);
GtkWidget* gtk_check_button_new_with_label (const gchar *label);
```

Für sie gilt dasselbe wie für die Funktionen des Toggle−Buttons.

Das Beispiel:

```
#include <gtk/gtk.h>
void togg(GtkWidget *widget, gpointer *data){
   if (gtk_toggle_button_get_active(GTK_TOGGLE_BUTTON(data)))
       g_print("State is 1\n");
   else
    g_print("State is 0\n");
}

int main( int argc,char *argv[] )
{

    GtkWidget *window;
    GtkWidget *button;

    gtk_init (&argc, &argv);

    /* Create a new window */
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "Check Button");

    /* Connect destroy event to the window. */
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

    /* Creates a check button */
    button=gtk_check_button_new_with_label("I'm a check button");

    /* Add the button to window */
    gtk_container_add(GTK_CONTAINER(window),button);

    /* Connect "toggled" event to the button */
    gtk_signal_connect (GTK_OBJECT (button), "toggled",
                        GTK_SIGNAL_FUNC(togg), (gpointer *)button);
    gtk_widget_show(button);
    gtk_widget_show (window);

    gtk_main ();
    return(0);
}
```

Labels

Mit Labels kannst du einen Text an eine beliebige Stelle im Fenster plazieren.
Um ein Label zu erzeugen, nimm einfach folgende Funktion:

```
GtkWidget* gtk_label_new(const gchar *text);
```

Mit der Funktion

```
gtk_label_set_text(GtkLabel *label, gchar *text);
```

kannst du den Text eines Labels jederzeit ändern.

```
gtk_label_set_justify(GtkLabel *label, GtkJustification jtype);
```

Die Funktion gtk_set_justify richtet den Text auf dem Label aus. *jtype* kann sein:

- GTK_JUSTIFY_LEFT, um den Text links auszurichten,
- GTK_JUSTIFY_RIGHT, um den Text rechts auszurichten,
- GTK_JUSTIFY_CENTER, um den Text zu zentrieren
- GTK_JUSTIFY_FILL, um den Text das ganze Label bedecken zu lassen.

```
gtk_label_set_line_wrap (GtkLabel *label,gboolean wrap);
```

Diese Funktion wird benutzt, um den Text in viele Teile aufzubrechen, wenn der Text länger ist als der Platz, auf den er passen soll. Wenn *wrap* 1 ist, wird der Text auf die nächste Zeile umgebrochen, andernfalls nicht.

```
gtk_label_get(GtkLabel *label, gchar **str)
```

Damit bekommst du den Text auf dem Label in die Variable *str*.

Tooltips

Ein Tooltip ist ein Text, der erscheint, wenn die Maus auf einem Bedienelement verweilt. Zum Beispiel kannst du einen Tip für einen Button setzen, damit der Text „verschicke diese Information“ erscheint, wenn sich die Maus über dem Button befindet.

Dafür muß zuerst ein GtkToolTips−Element erzeugt werden:

```
GtkToolTips* gtk_tooltips_new();
```

Dann wird dieser Tooltip einem Element angeheftet:

```
gtk_tooltips_set_tip(GtkTooltips *tooltips, GtkWidget *widget,
                     const gchar *tip_text,const gchar *tip_private);
```

Ein kleines Beispiel:

```
#include <gtk/gtk.h>
int main( int argc,char *argv[] )
{
    GtkWidget *window;
    GtkWidget *button;
    GtkTooltips *tip;

    gtk_init (&argc, &argv);

    /* Create a new window */
    window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title (GTK_WINDOW (window), "Tooltips");

    /* Connect destroy event to the window. */
    gtk_signal_connect (GTK_OBJECT (window), "destroy",
                        GTK_SIGNAL_FUNC (gtk_main_quit), NULL);

    /* Creates a button */
```

```
    button=gtk_button_new_with_label("I'm a Button");

    /* Add the button to window */
    gtk_container_add(GTK_CONTAINER(window),button);

    /* Creates a tooltips*/
    tip=gtk_tooltips_new();

    /* Attache this tooltips to button with text*/
    gtk_tooltips_set_tip(tip, button, "Click me!",NULL);

    gtk_widget_show(button);
    gtk_widget_show (window);

    gtk_main ();
    return(0);
}
```

Einige andere Funktionen:

```
gtk_tooltips_enable (GtkTooltips *tooltips);
```

Schaltet die Tooltips ein.

```
gtk_tooltips_disable (GtkTooltips *tooltips);
```

Schaltet die Tooltips aus.

Um die Tooltip−Daten eines Elements zu bekommen, brauchen wir

```
GtkTooltipsData* gtk_tooltips_get_data(GtkWidget *widget);
```

GtkTooltipsData ist ein solcher *struct*:

```
struct _GtkTooltipsData
{
   GtkTooltips *tooltips;
   GtkWidget *widget;
   gchar *tip_text;
   gchar *tip_private;
   GdkFont *font;
   gint width;
   GList *row;
};
```

Um die Verzögerung festzulegen, mit der der Text erscheint:

```
gtk_tooltips_set_delay (GtkTooltips *tip, guint delay)
```

Die Combobox

Eine Combobox ist eine Textbox, die vorgefertigte, auswählbare Einträge in einer Pull−down−Liste anbietet.

Eine Combobox kann erzeugt werden durch

```
GtkWidget *gtk_combo_new();
```

Und wir brauchen eine Liste von Optionen in Form eines *GList struct*.

```
GList *glist=NULL;
```

Eine Option kann der Liste hinzugefügt werden mit

```
GList *g_list_append(GList *list, gchar *option);
```

Dann muß diese Liste der Combobox zugeordnet werden:

```
gtk_combo_set_popdown_strings(GtkCombo *combo, GList *List);
```

Die Combobox ist fertig. Um die ausgewählte Option zu erfahren, benutze:

```
gchar *gtk_entry_get_text(GtkEntry *entry);
```

entry ist in diesem Fall `GTK_ENTRY(GTK_COMBO(combo)->entry))`.

```
gtk_combo_set_use_arrows(GtkCombo *combo,gint val);
```

Diese Funktion wird benutzt, um die Hoch/runter–Pfeiltasten auf der Tastatur zur Steuerung der Combobox ein– oder auszuschalten. Wenn *val* 0 ist, funktionieren diese Tasten nicht, ansonsten ändern sie den Wert. Aber wenn der Wert in der Combobox von den Werten in der Liste abweicht, **funktionieren diese Tasten nicht**.

```
gtk_combo_set_use_arrows_always(GtkCombo *combo,gint val);
```

Diese Funktion leistet dasselbe wie *gtk_combo_set_use_arrows*, aber wenn der Wert in der Combobox von den Werten in der Liste abweicht, **funktionieren die Tasten**.

```
gtk_combo_set_value_in_list     (GtkCombo *combo,
                                 gboolean val, gboolean ok_if_empty);
```

Wenn *val* 1 ist, kannst du einen Wert in die Liste eingeben. Wenn *ok_if_empty* 1 ist, darf der Wert leer sein.

```
#include <gtk/gtk.h>
void act(GtkWidget *widget, gpointer *data){
   g_print((gchar *)data);
}

int main( int argc,char *argv[] ) {
   GtkWidget *window;
   GtkWidget *combo;
   GtkWidget *button;
   GtkWidget *box;
   GList *list=NULL;

   list=g_list_append(list,"Slackware");
   list=g_list_append(list,"RedHat");
   list=g_list_append(list,"SuSE");

   gtk_init (&argc, &argv);

   /* Create a new window */
   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
   gtk_window_set_title (GTK_WINDOW (window), "Combo Box");

   /* Connect destroy event to the window. */
```

```
gtk_signal_connect (GTK_OBJECT (window), "destroy",
                    GTK_SIGNAL_FUNC(gtk_main_quit), NULL);

/* Create a new horizontal box */
box=gtk_hbox_new(1,0);
gtk_container_add(GTK_CONTAINER(window),box);

/* Creates a combo box */
combo=gtk_combo_new();

/* Sets the list */
gtk_combo_set_popdown_strings(GTK_COMBO(combo),list);

/* Enables up/down keys change the value. */
gtk_combo_set_use_arrows_always(GTK_COMBO(combo),1);

gtk_box_pack_start(GTK_BOX(box), combo,1,1,1);

button=gtk_button_new_with_label("Write it");
gtk_signal_connect(GTK_OBJECT(button),"clicked",GTK_SIGNAL_FUNC(act),
          gtk_entry_get_text(GTK_ENTRY(GTK_COMBO(combo)->entry)));
gtk_box_pack_start(GTK_BOX(box), button,1,1,1);

gtk_widget_show(box);
gtk_widget_show(combo);
gtk_widget_show(button);
gtk_widget_show (window);

gtk_main ();
return(0);
}
```

Alle Kommentare sind willkommen.
