1 GTK+

GTK+ ist freie Software

- Frei wie in Freiheit, nicht wie in Freibier

FSF Definition – 4 Freiheiten

- das Programm für jeden Zweck ausführen
- den Quellcode lesen und verändern
- das Programm weiter geben
- eine veränderte Version des Programms weiter geben

1.1 Historie

1995	Peter Mattis, Spencer Kimball **GIMP** General Image Manipulation Program link	
1996	**GIMP 0.54** erste offizielle Version	Matthias Ettrich **KDE** Kool Desktop Environment link
1997	Miguel de Icaza **GNOME** GNU Network Object Model Environment link	
1998	**GTK+ 1.0** erste offizielle Version	**KDE 1.0** erste offizielle Version
1999	**GNOME 1.0** erste offizielle Version	**QT 2.0** unter neuer Lizenz: QPL link
2000		**QT 2.2** für GNU/Linux unter GPL (2003 und 2005 auch für Mac-OS-X, bzw. Windows)
2002	**GTK+ 2.0** **GNOME 2.0**	
???	**GTK+ 3.0**	

1.2 G-Familie

1.2.1 wichtigste Mitglieder

GLib

- Datentypen
- Datenstrukturen

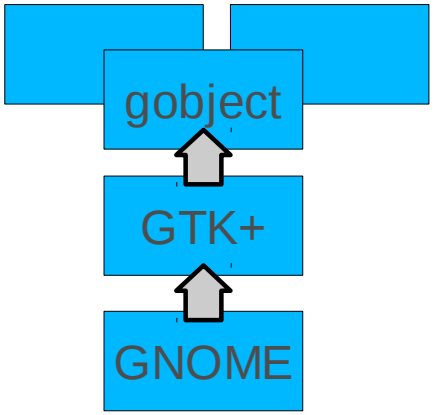
GObject

- dynamisches Typsystem
- (pseudo) Objekte

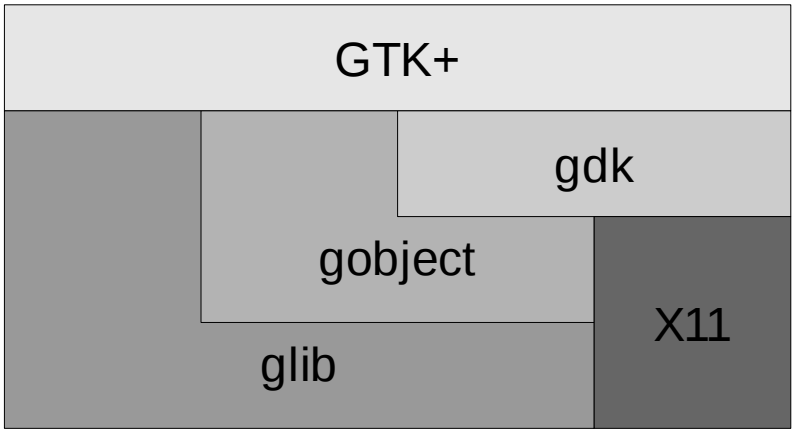
GTK+

- Widget Toolkit

Code-Wanderung



Aufbau

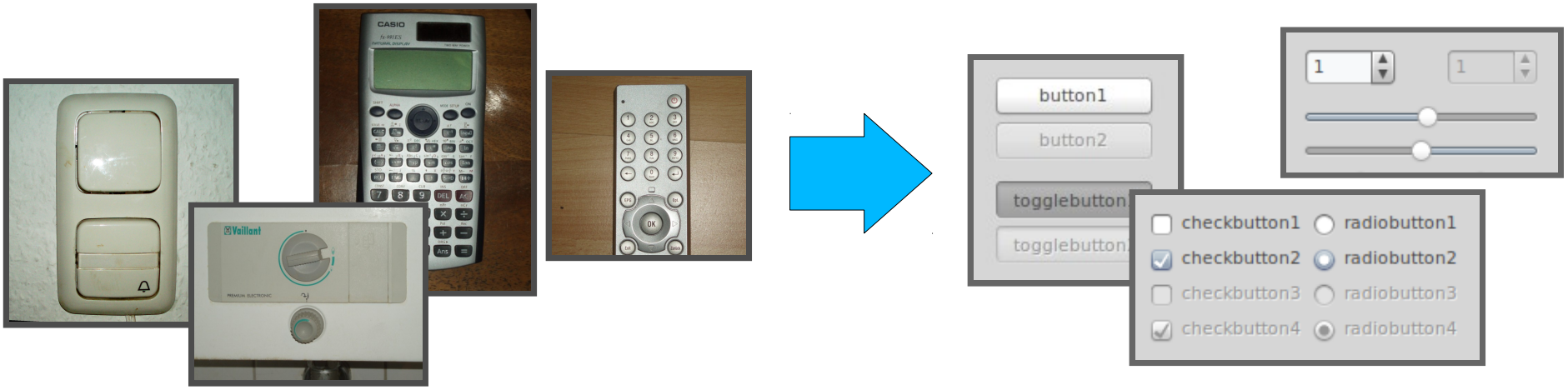


2 GUI Entwicklung

Motivation

- „vereinfachte“ Bedienung der Maschine Computer

Analogien



Ereignisgesteuerte Programmierung

- Kontrollfluss nicht fest vorgegeben
- Arbeitsweise asynchron

2.1 Terminalprogrammierung

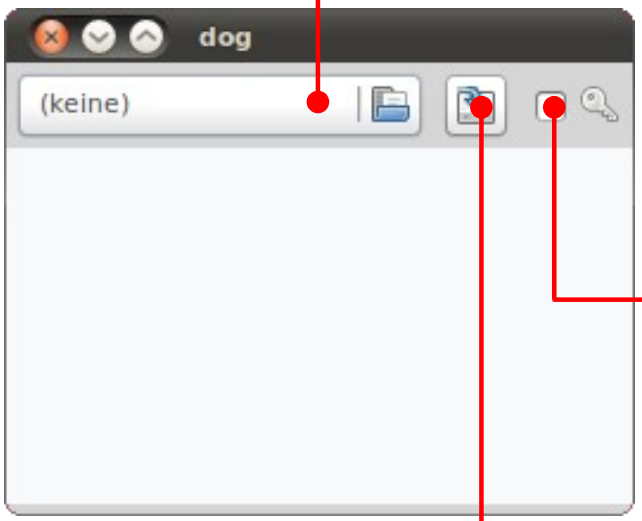
```
$ ./kitty <<EOF > hi
> Hi Kitty!
> EOF
$
```

```
$ ./kitty <<EOF > message
> Na, lebst Du etwa immer noch?
> Sind ja eine ganze Menge, sieben Leben ...
> EOF
$
```

```
$ ./kitty hi message
Hi Kitty!
Na, lebst Du etwa immer noch?
Sind ja eine ganze Menge, sieben Leben ...
$
```

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  FILE *fp = stdin;
  int c;

  while (*argv++ !=NULL)
    {
      if ((argc != 1)
          && ((fp=fopen (*(argv), "r")) == NULL))
        continue;
      while ((c=fgetc (fp)) != EOF)
        putchar (c);
      fclose (fp);
    }

  return 0;
}
```

2.2 GUI Programmierung

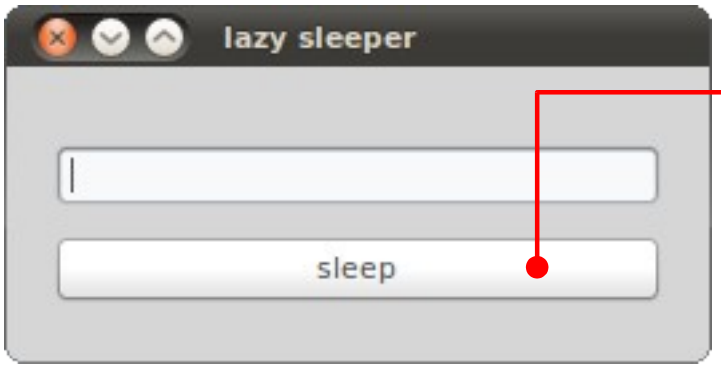



```
void on_open_button_file_set (GtkWidget* widget, gpointer data)
{
  gchar* text;

  g_free (filename);
  filename = gtk_file_chooser_get_filename (GTK_FILE_CHOOSER (widget));
  g_file_get_contents (filename, &text, NULL, NULL);
  g_object_set (buffer, "text", text, NULL);
  g_free (text);
}
```

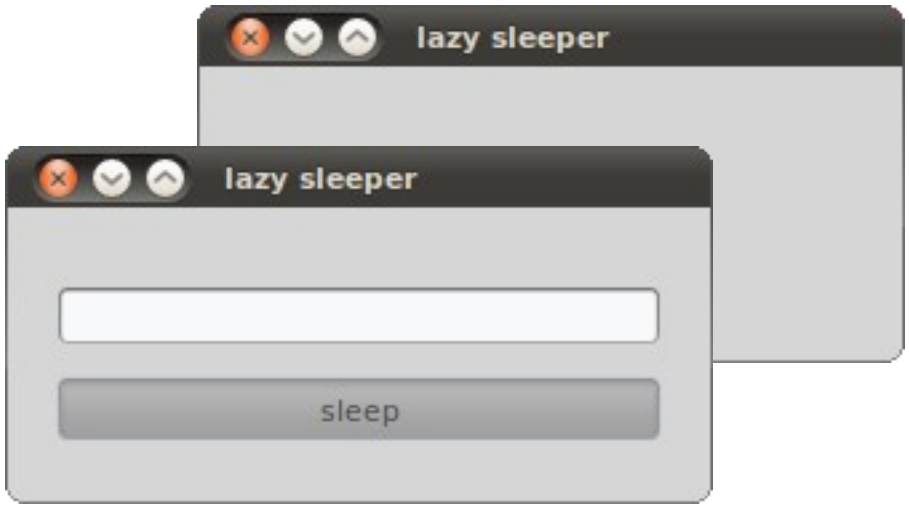
```
void on_lock_toggled (GtkWidget* widget, gpointer data)
{
  gboolean state;
  g_object_get (widget, "active", &state, NULL);
  g_object_set (view, "editable", !state, NULL);
}
```

```
void on_save_clicked (GtkWidget* chooser, gpointer data)
{
  gchar* text;

  g_object_get (buffer, "text", &text, NULL);
  g_file_set_contents (filename, text, -1, NULL);
}
```

2.3 Aber Vorsicht …




```
void button_clicked (GtkWidget *widget, gpointer data)
{
  sleep (10);
}
```



3 Konzepte

Allgemein

- Objektorientierung
- Steuerelemente
  - Erscheinungsbild
  - Verhalten
- Layout-Management

GTK+ spezifisch

- Properties
  - legen das Erscheinungsbild eines Steuerelements fest
- Signale
  - legen das Verhalten eines Steuerelements fest
- Behälter
  - richten verschiedene Steurelemente zueinander aus

3.1 Objekte

Typ

- GtkButton

Typkennung

- GTK_TYPE_BUTTON

Typecast-Makro

- GTK_BUTTON (...)

Methoden

- gtk_button_...

3.1.1 Erzeugung

klassisch

- gtk_button_new ()
- gtk_button_new_with_label ("Button")
- gtk_button_new_with_mnemonic ("_Button")
- gtk_button_new_from_stock (GTK_STOCK_CANCEL)

gobject

- g_object_new (GTK_TYPE_BUTTON, …, NULL)

automatisch

- ~~libglade~~
- GtkBuilder

3.1.2 Properties

GtkButton* button;

Klassisch

- gtk_button_set_focus_on_click (button, TRUE)
- bool = gtk_button_get_focus_on_click (button)
- ...

gobject

- g_object_set (button, ..., NULL)
- g_object_get (button, …, NULL)

"focus-on-click"	gboolean
"image"	GtkWidget*
"image-position"	GtkPositionType
"label"	gchar*
"relief"	GtkReliefStyle
"use-stock"	gboolean
"use-underline"	gboolean
"xalign"	gfloat
"yalign"	gfloat

3.2 Signale

GtkButton* button;

"activate"
"clicked"
"enter"
"leave"
"pressed"
"released"

manuell

- g_signal_connect (button, "clicked", G_CALLBACK (button_clicked), NULL)

automatisch

- GtkBuilder

4 Layout-Management

GtkWidget *box = g_object_new (GTK_TYPE_HBOX, NULL)

Container

- Fentster
- Boxen
- Tabellen

einfügen

- gtk_container_add (GTK_CONTAINER (box), widget)

child properties

- gtk_..._child_set_property (container, child, …, NULL)
- gtk_..._child_get_property (container, child, …, NULL)

"expand"	gboolean
"fill"	gboolean
"pack-type"	GtkPackType
"padding"	guint
"position"	gint

5 GtkBuilder

GtkBuilder *builder = g_object_new (GTK_TYPE_BUILDER, NULL)

Einlesen

- gtk_builder_add_from_file (builder, "ui.xml", NULL)

Signale

- gtk_builder_connect_signals (builder, NULL)

Referenzen

- GtkWidget *window = GTK_WIDGET (gtk_builder_get_object (builder, "window"))
- ….

g_object_unref (G_OBJECT(builder));

6 Hilfe zur Selbsthilfe

gtk-demo

- Widgets
- Quellcode

devhelp

- Referenzen
- Suchfunktion

gtk.org

- online HTML Dokumentation