COMP 2400 UNIX Tools

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

GTK+

- GTK+ = Gimp Tool Kit, GIMP = GNU Image Manipulation Program
- Basis for Gnome
- Written in C, bindings for many other languages
- Available for most platforms (Linux, BSD, Solaris, OS X, Win32)
- Programming model: single-threaded event-driven MVC

GTK+

- Written in C with C API
- Has objects, methods and inheritance!
- Uses explicit reference counting for memory management
- `glib` provides type system
- Convention for methods: `type_name_method_name(recei arguments)`

MVC

Model The application's central data structure

View Graphical representation of the data

Controller Code controlling modifications to data and selection of views

Views

- Multiple views can render the same data
- Choice of view depends on data and user
- GUI toolkit determines available views
- In GTK+, all views are subtypes of `GtkWidget`

Common Views

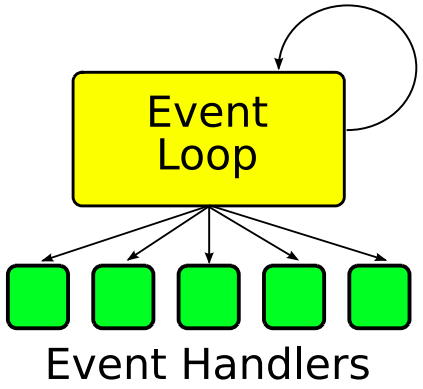
- Progress bar
- Image
- Entry line
- Spin button
- Scroll bar
- Toggle and check buttons

Creating Views

```
GtkWidget * vbox = gtk_vbox_new(FALSE, 0);
GtkWidget * ybutton = gtk_button_new_with_label("yes");
GtkWidget * nbutton = gtk_button_new_with_label("no");
gtk_box_pack_start(GTK_BOX(vbox), ybutton, FALSE, FALSE);
gtk_box_pack_start(GTK_BOX(vbox), nbutton, FALSE, FALSE);
GtkWidget * window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
gtk_window_set_title(GTK_WINDOW(window), "Hello World!");
gtk_container_add(GTK_CONTAINER(window), vbox);
```

Event Driven Programming

- One execution stream
- Register interest in events
- Event loop waits for events
- Event loop invokes callbacks
- Event handlers must be short-lived

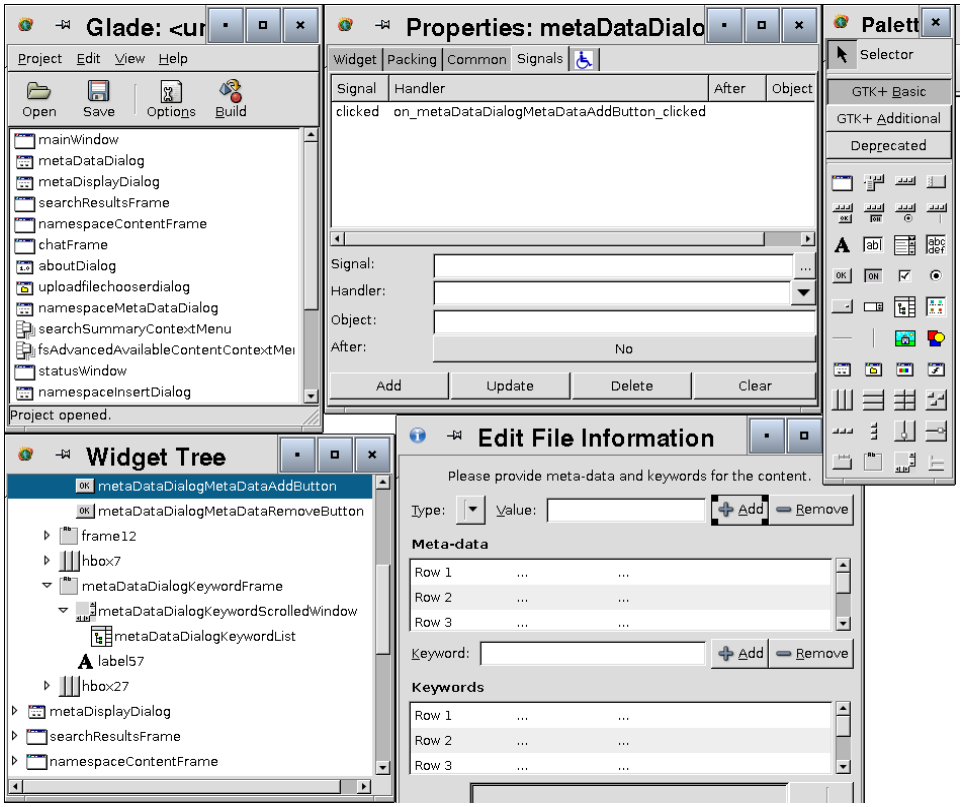


Hollywood Principle: "Don't call us, we'll call you..." – Dafydd Rees

GTK Hello World

```
##include <gtk/gtk.h>
int main (int argc, char ** argv) {
  GtkWidget * win, * but;
  gtk_init(&argc, &argv );
  win = gtk_window_new(GTK_WINDOW_TOPLEVEL);
  g_signal_connect(win, "destroy",
                   G_CALLBACK(gtk_main_quit), NULL);
  but = gtk_button_new_with_label("Close!");
  g_signal_connect_swapped(but, "clicked",
                   G_CALLBACK(gtk_object_destroy), win);
  gtk_container_add(GTK_CONTAINER(win), but);
  gtk_widget_show_all(win);
  gtk_main();
  return 0;
}
```

Glade


Glade: <ur
Project Edit View Help
Open Save Options Build
mainWindow
metaDataDialog
metaDisplayDialog
searchResultsFrame
namespaceContentFrame
chatFrame
aboutDialog
uploadfilechooserdialog
namespaceMetaDataDialog
searchSummaryContextMenu
fsAdvancedAvailableContentContextMe
statusWindow
namespaceInsertDialog
Project opened.
Properties: metaDataDialo
Widget Packing Common Signals
Signal Handler After Object
clicked on_metaDataDialogMetaDataAddButton_clicked
Signal:
Handler:
Object:
After:
No
Add Update Delete Clear
Palett
Selector
GTK+ Basic
GTK+ Additional
Deprecated
Widget Tree
metaDataDialogMetaDataAddButton
metaDataDialogMetaDataRemoveButton
frame12
hbox7
metaDataDialogKeywordFrame
metaDataDialogKeywordScrolledWindow
metaDataDialogKeywordList
label57
hbox27
metaDisplayDialog
searchResultsFrame
namespaceContentFrame
Edit File Information
Please provide meta-data and keywords for the content.
Type: Value: Add Remove
Meta-data
Row 1 ... ...
Row 2 ... ...
Row 3 ... ...
Keyword: Add Remove
Keywords
Row 1 ... ...
Row 2 ... ...
Row 3 ... ...

Glade

```
#include <gtk/gtk.h>
#include <glade/glade.h>
void some_handler(GtkWidget *widget) { }
int main(int argc, char **argv) {
  GladeXML *xml;
  GtkWidget *window;

  gtk_init(&argc, &argv);
  xml = glade_xml_new("filename.glade", NULL, NULL);
  window = glade_xml_get_widget(xml, "widgetname");
  glade_xml_signal_autoconnect(xml);
  gtk_widget_show_all(window);
  gtk_main();
  return 0;
}
```

Glade

An `autoconf` setup for GTK+ and glade is part of

`https://gnunet.org/svn/gnunet-gtk/`

GtkTreeView

- The most complex Widget
- Related classes: `GtkTreeModel`, `GtkTreeStore`, `GtkListStore`, `GtkTreePath`, GtkCellRenderer, `GtkTreeIter`
- Glade support is somewhat lacking

Tutorial: `http://scentric.net/tutorial/`

GtkTreeModel

- Essentially an $n \times m$ matrix of data
- Constant number of colums with types specified at creation time
- Rows have a tree structure
- This is just an interface!

GtkListStore

```
enum { COL_NAME = 0, COL_AGE, NUM_COLS };

static GtkTreeModel * create_and_fill_model() {
  GtkListStore  *store;
  GtkTreeIter    iter;
  store = gtk_list_store_new(NUM_COLS, G_TYPE_STRING, G_TYPE_UINT);
  gtk_list_store_append (store, &iter);
  gtk_list_store_set (store, &iter,
                      COL_NAME, "Heinz El-Mann",
                      COL_AGE, 51,
                      -1);
  gtk_list_store_append (store, &iter);
  gtk_list_store_set (store, &iter,
                      COL_NAME, "Jane Doe",
                      COL_AGE, 23,
                      -1);
  return GTK_TREE_MODEL (store);
}
```

GtkCellRenderer

```
static GtkWidget * create_view_and_model() {
  GtkCellRenderer * renderer;
  GtkTreeModel    * model;
  GtkWidget       * view;
  view = gtk_tree_view_new();
  renderer = gtk_cell_renderer_text_new();
  gtk_tree_view_insert_column_with_attributes
   (GTK_TREE_VIEW(view), -1, "Name", renderer,
    "text", COL_NAME, NULL);
  renderer = gtk_cell_renderer_text_new();
  gtk_tree_view_insert_column_with_attributes
   (GTK_TREE_VIEW(view), -1, "Age",  renderer,
    "text", COL_AGE,  NULL);
  model = create_and_fill_model();
  gtk_tree_view_set_model(GTK_TREE_VIEW(view), model);
  g_object_unref(model);
  return view;
}
```

Questions

?