Helicon technologies Computerized measurement systems

16 Hapoel str. Nof-Yam Herzeliya 46625 ISRAEL

GTK+ GUI programming

Ori Idan

Helicon technologies

What is it?

- GUI library written in C with bindings for many other languages
- LGPL License
- Part of the GNU project
- Initially developed for and used by the GIMP
- Used today as the basis of GNOME
- Portings to other operating systems including MS-Windows

Language bindings

- Language bindings enable you to write GTK+ programs using languages other then C.
- GTK+ was written with language bindings in mind from the very beginning
- Officialy supported languages: C++, Java, Perl
- Other languages supported: PHP, Python, Ruby, TCL, Eifel, C#, ADA, Lisp and many others.

Architecture and components

- Fully object oriented although written in C.
- Uses classes and callback functions implemented as structures and pointers to functions
- Contains the following libraries:
  - Glib – low level core library functions providing event loop, threads, dynamic loading, object system, string and list manipulation etc.
  - Pango – Layout and rendering of text with emphasis on I18N
  - ATK – Accessibility toolkit supporting screen readers and alternative input devices.

Show me the code...


```
int main(int argc, char *argv[]) {
    GtkWidget *window, *button;
    /* initialize library and parse command line arguments */
    gtk_init(&argc, &argv);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    g_signal_connect(G_OBJECT(window),"delete_event",
      G_CALLBACK(delete_event), NULL);
    g_signal_connect(G_OBJECT(window), "destroy",
      G_CALLBACK(destroy), NULL);
    gtk_container_set_border_width(GTK_CONTAINER(window), 10);
    button = gtk_button_new_with_label("Hello world");
    gtk_container_add (GTK_CONTAINER (window), button);
    gtk_widget_show(button);
    gtk_widget_show(window);
    gtk_main();
    return 0;
}
```

Show me the code...


```
gboolean delete_event(GtkWidget *widget, GdkEvent *event,
   gpointer data) {
    printf("delete event occured\n");

    return FALSE;

}

void destroy(GtkWidget *widget, GdkEvent *event,
   gpointer data) {

    gtk_main_quit();

}
```

How to compile

Use the pkg-config utility

```
gcc -o hello hello.c `pkg-config --cflags --libs gtk+2.0`
```

pkg-config will give all the flags needed to compile using this library.

Signals

- Signals are used to relate actions to mouse clicks, mouse moves, keyboard, timer etc.
- Each widget can have several signal handlers for different signals or for same signal
- The g_signal_connect() function connects a signal to a callback function to handle the signal.
- Type of callback function depends on the signal.

More code...

First we create a callback function to be called when clicking the button:

```
void ButtonCallback(GtkWidget *button, gpointer
        *data) {
   printf("Button pressed\n");
   gtk_main_quit();
}
```

Then we connect the function to the signal:

```
g_signal_connect(G_OBJECT(button), "clicked",
      G_CALLBACK(ButtonCallback), NULL);
```

Now when we run the program, pressing the button will print a message and quit.

Packing widgets

- Widgets are packed in boxes and tables.
- There are two types of boxes VBOX and HBOX
- Table can be regarded as combination of HBOX and VBOX
- Each cell may contain either a box or one widget
- Create a box using gtk_vbox_new or gtk_hbox_new
- Add the box using gtk_container_add
- Create a table using gtk_table_new

Packing widgets example

- Add two buttons in a window:

```
box1 = gtk_hbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (window), box1);

/* create a button and add it to box */
button = gtk_button_new_with_label ("Button 1");
gtk_box_pack_start (GTK_BOX(box1), button, TRUE,
      TRUE, 0);


/* create another button and add it to box */
button = gtk_button_new_with_label ("Button 2");
gtk_box_pack_start(GTK_BOX (box1), button, TRUE,
TRUE, 0);
```

Trees and lists

- Use the GtkTreeView
- There is also the GtkList widget but it is deprecated and should not be used
- Tree view can be used for both trees and lists
- The idea is a seperation between a view and a model
- There are two built in models, list and tree
- List have columns and rows

GtkTreeView list model

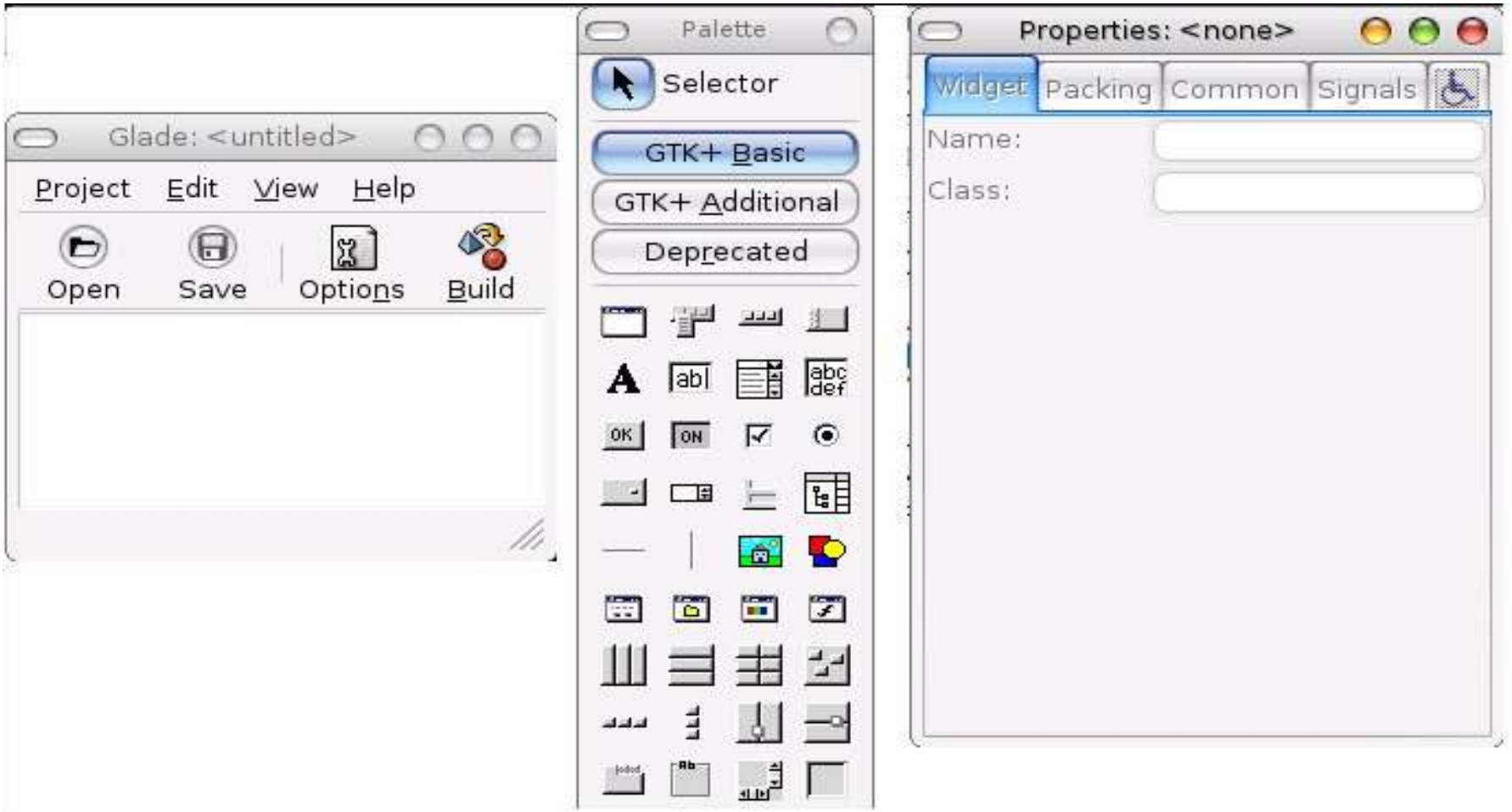
- Each column in the list model should be mapped to a column on the GtkTreeView widget
- Unmaped columns are not shown and use to store information that is not to be seen by the user such as record number etc.
- Mapping column is done by the create column function

List example

```
list = gtk_list_store_new(2, G_TYPE_STRING, G_TYPE_STRING);

gtk_tree_view_set_model(GTK_TREE_VIEW(TreeView),
   GTK_TREE_MODEL(list));

/* Now add the columns to the list */
renderer = gtk_cell_renderer_text_new();

g_object_set(G_OBJECT(renderer), "foreground", "black",
   NULL);

column = gtk_tree_view_column_new_with_attributes("מספר
   ראשי" ,renderer, "text", 0, NULL);

gtk_tree_view_append_column(GTK_TREE_VIEW(TreeView), column);

column = gtk_tree_view_column_new_with_attributes("מספר
   שלם" , renderer, "text", 1, NULL);

gtk_tree_view_append_column(GTK_TREE_VIEW(TreeView), column);
```

Building user interfaces

The GLADE utility is a GUI for user interfaces.



What about IDE?

- Anjuta is a great IDE for GTK+
- It can create GTK+ and GNOME projects
- It can be used with both C and C++ or other languages
- Integrates with GLADE
- Uses gcc and gdb for compiling and debugging

What about localization

- GTK+ supports localization
- Works with GNU gettext
- Supports hebrew and RTL from version 2.
- Hebrew support includes menus from right to left and packing HBOX and tables from right to left

How to learn

- Start with the GTK+ Tutorial at: http://www.gtk.org
- Use the online reference manual
- Look at other software packages such as gedit etc.

After all this is what open source is about...

Thank you...

Questions ???

Ori Idan: ori@helicontech.co.il
http://www.helicontech.co.il