Beginning GTK+ Programming

KLDPConf

17 Apr 2004

Dongsu Jang <jdongsu@pyunji.net>

Contents

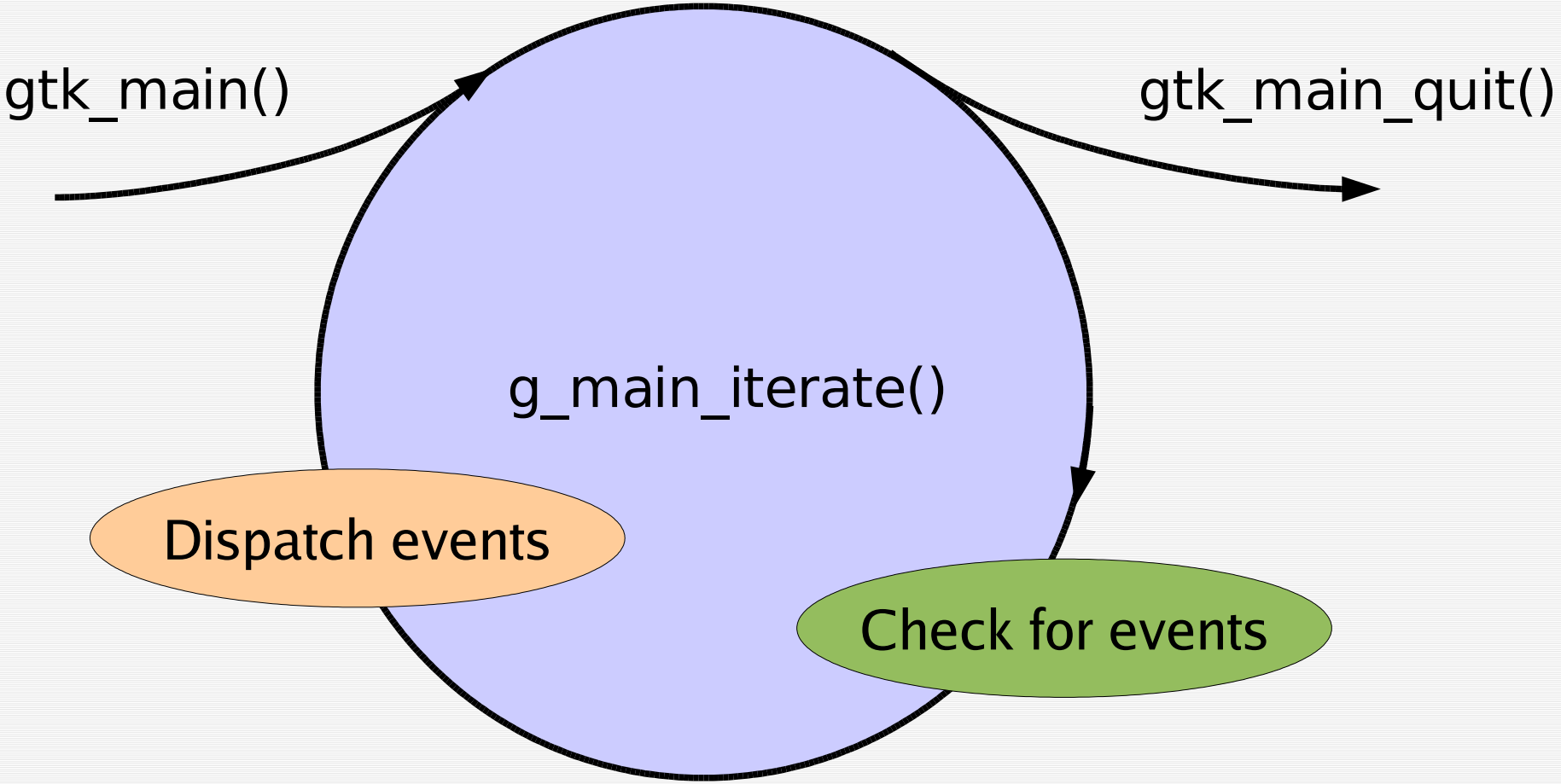
Getting Started

- Hello,GTK+!
- The Main Event Loop
- Event, Signal and Callback
- Compile & Run
- ./configure;make;make install?
- Glade GUI Builder
- Hello,Glade! - libglade
- i18n
- 안녕, 글레이드! - gettext

Hello,GTK+!


```
#include <gtk/gtk.h>
...
int main(int argc, char **argv)
{
    GtkWidget *window, *button;
    gtk_init(&argc, &argv);
    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    g_signal_connect(window, "destroy",
                             G_CALLBACK(gtk_main_quit), NULL);
    button = gtk_button_new_with_label("Hello, GTK+!");
    g_signal_connect(button, "clicked",
                           G_CALLBACK(hello_clicked), window);
    gtk_container_add(GTK_CONTAINER(window), button);
    gtk_widget_show_all(window);
    gtk_main();
    return 0;
}
```

The Main Event Loop



Event, Signal and Callback (1)

- Event Driven Programming:
  - After initial setup, program waits for “events” a nd reacts for them.
- gtk_main() starts event loop and runs until gtk_ main_quit() is called.
- Low-level events(button presses, keystrokes) h andled by GTK+ and converted into high-level signals.
- Signal notification is essential part of GUI
  - Need to find out when, e.g., a button is clicked.
- GTK+ runs all handlers for a specific object and signal.

Event, Signal and Callback (1)

- **connect** a **callback** function to **signal**:

```
static void hello_clicked(GtkButton *button,
                                        gpointer user_data)
{
    GtkWidget *dialog =
     gtk_message_dialog_new(GTK_WINDOW(user_data),
                  GTK_DIALOG_DESTROY_WITH_PARENT,
                  GTK_MESSAGE_INFO, GTK_BUTTONS_OK,
                  "Hello,GTK+!");
    gtk_dialog_run(GTK_DIALOG(dialog));
    gtk_widget_destroy(dialog);
}

g_signal_connect(button, "clicked",
                         G_CALLBACK(hello_clicked), window);
```

Event, Signal and Callback (2)

- Set of signals associated with each class
  - Button has "pressed", "released", "clicked", ...
  - Also inherits "focus_in_event", "focus_out_even t" from GtkWidget, etc, etc.
- Differerent signals, different signatures:

```
void clicked_cb        (GtkButton          *GtkButton,
                        gpointer            user_data);
void row_activated_cb (GtkTreeView        *tree_view,
                        GtkTreePath        *path,
                        GtkTreeViewColumn *column,
                        gpointer            user_data);
```

  - Listed in reference documentation
  - User data always last argument
  - Search “ in DevHelp ;^)

Event, Signal and Callback (3)

- Some signals have boolean return values
  - ex. "delete_**event**", emitted when user clicks o n close button in title bar:

```
gboolean delete_event_cb (GtkWidget   *widget,
                          GdkEventAny *event,
                          gpointer     user_data);
```

- Convention is TRUE return stops signal emi ssion
  - TRUE: I handled it, don't do anything more
  - FALSE: I didn't handle it, do whatever you woul d normally do. (Here, destroy the window)

Compile & Run

- pkg-config command line utility gives proper compiler, linker flags.
  - Note backticks ``

```
$ gcc `pkg-config --cflags --libs gtk+-2.0` -
  g -Wall -o hello hello.c
$ ./hello
```

  - Behind the scenes:

```
$ pkg-config --cflags gtk+-2.0
-I/usr/include/gtk-2.0 -I/usr/lib/gtk-2.0/include -I/usr/inc
  lude/atk-1.0 -I/usr/include/pango-1.0 -I/usr/X11R6/include
  -I/usr/include/freetype2 -I/usr/include/freetype2/config -
  I/usr/include/glib-2.0 -I/usr/lib/glib-2.0/include
$ pkg-config --libs gtk+-2.0
-Wl,--export-dynamic -lgtk-x11-2.0 -lgdk-x11-2.0 -latk-1.0 -
  lgdk_pixbuf-2.0 -lm -lpangoxft-1.0 -lpangox-1.0 -lpango-1.
  0 -lgobject-2.0 -lgmodule-2.0 -ldl -lglib-2.0
```

./configure;make;make install? (1)

- What is GNU Build Tools?
  - SeeAlso KLDPWiki:AutoTools
  - SeeAlso GnomeKorea:AutoTools
- Setup Work Directory:

```
$ mkdir hello2
$ cd hello2
$ mkdir src
$ mkdir pixmaps
$ cp ../hello1/hello.c src/main.c
$ cp /usr/share/pixmaps/gimp.png pixmaps/hello.png
$ tree
.
|-- pixmaps
|   `-- hello.png
`-- src
    `-- main.c
```

./configure;make;make install? (2)

- Create configure.in:

```
AC_INIT(src/main.c)
AM_CONFIG_HEADER(config.h)
PACKAGE=hello
VERSION=0.2
AM_INIT_AUTOMAKE($PACKAGE,$VERSION)
AC_PROG_CC
AC_PROG_INSTALL
AC_STDC_HEADERS
PKG_CHECK_MODULES(DEPS, gtk+-2.0 >= 2.0)
AC_SUBST(DEPS_CFLAGS)
AC_SUBST(DEPS_LIBS)
AC_OUTPUT([
Makefile
src/Makefile
pixmaps/Makefile
])
```

./configure;make;make install? (3)

- Create Makefile.am:

```
SUBDIRS = src pixmaps
```

- Create src/Makefile.am:

```
bin_PROGRAMS = hello
hello_SOURCES = main.c
hello_LDADD = $(DEPS_LIBS)
AM_CPPFLAGS = $(DEPS_CFLAGS) \
     -DPIXMAPS_DIR=\""$(datadir)/pixmaps"\"
```

- Create pixmaps/Makefile.am:

```
pixmapsdir = $(datadir)/pixmaps
pixmaps_DATA = hello.png
EXTRA_DIST = $(pixmaps_DATA)
```

./configure;make;make install? (4)

- Modify src/main.c:

```
#include “config.h”
gtk_window_set_default_icon_from_file(
                                PIXMAPS_DIR "/hello.png", NULL);
```

- Bootstrapping:

```
$ tree
.
|-- Makefile.am
|-- configure.in
|-- pixmaps
|   |-- Makefile.am
|   `-- hello.png
`-- src
    |-- Makefile.am
    `-- main.c
$ aclocal;autoheader;touch stamp-h;autoconf;automake -a -c
```

./configure;make;make install? (5)

- Compile & Run!

```
$ configure --prefix=$HOME/test
$ make
$ make install
$ PATH=$HOME/test/bin:$PATH hello
```

- Make a distribution tarball

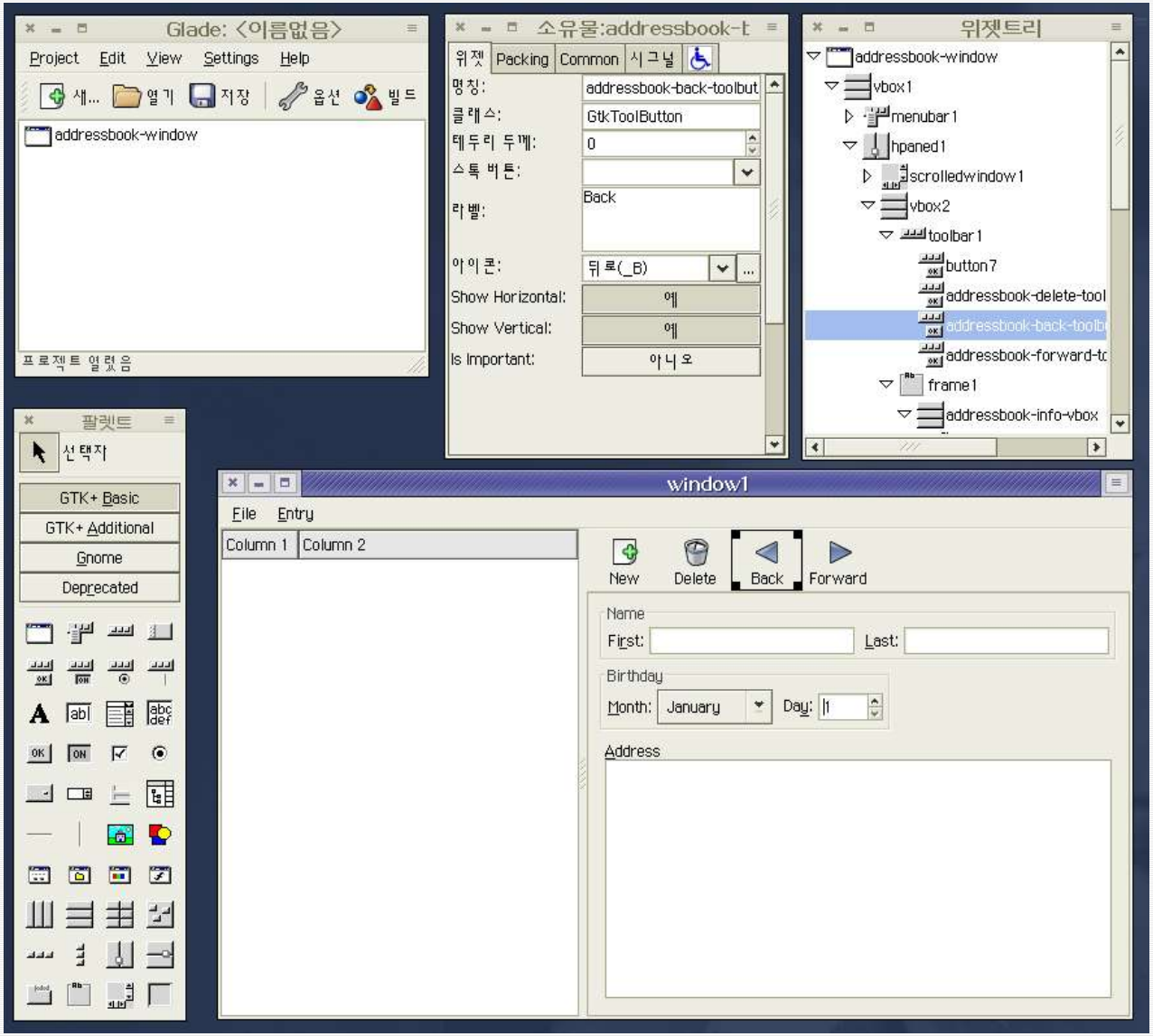
```
$ make dist
$ ls hello*
hello-0.2.tar.gz
```

- Create .cvsignore before cvs commit

```
Makefile
Makefile.in
aclocal.m4
autom4e.cache
stamp-h*
...
```

Glade GUI Builder

- WYSIWYG GUI editor for visual layouts
- Saves layouts as XML files(.glade)
- Generates source code(in C) and projects: DO **NOT** USE!



Hello,Glade! - libglade (1)

- Loads XML layouts into application

```
GladeXML *glade;
glade = glade_xml_new("hello.glade", "window1",NULL);
```

- Can specify callback names in Glade, have libglade look them them up in program

```
glade_xml_signal_autoconnect(glade);
```

- Functions need to be exported:
  - Have to be public, **not static**
  - `pkg-config --libs libglade` gives proper linker flags
- Provides pointers to widgets for furture referenc

```
GtkWidget* button1;
button1 = glade_xml_get_widget(glade, “button1”);
```

Hello,Glade! - libglade (2)

- Modify configure.in & */Makefile.am:

```
...
VERSION=0.3
...
PKG_CHECK_MODULES(DEPS, gtk+-2.0 >= 2.0 libglade-2.0 >= 2.0)
...
```

- Modify src/main.c

```
...
#include <libglade/libglade.h>
...
     GladeXML *glade;
...
     glade = glade_xml_new(GLADE_DIR "hello.glade", "windo
  w",NULL);
     glade_xml_signal_autoconnect(glade);
...
```

i18n

- What is i18n?
  - InternationalizatioN
  - SeeAlso l10n, a11y
- All text in GTK+ is:
  - Unicode charset(ISO-10646, 4 bytes per char)
  - UTF-8 encoding(1~4 bytes, ASCII compatible)
- Manipulate unicode with GLib
  - g_convert()
  - g_utf8_xxx()
  - **g_locale_to/from_utf8()**
  - **g_filename_to/from_utf8()**
- Provides translations with GNU **gettext**

안녕, 글레이드! - gettext (1)

- Modify src/main.c

```
...
#include <libintl.h>
#include <locale.h>
#define _(text) gettext(text)
...
                    _("Hello,Glade!"));
...
#ifdef ENABLE_NLS
    bindtextdomain(GETTEXT_PACKAGE, LOCALE_DIR);
    bind_textdomain_codeset(GETTEXT_PACKAGE, "UTF-8");
    textdomain(GETTEXT_PACKAGE);
#endif
...
```

안녕, 글레이드! - gettext (2)

- Modify configure.in:

```
...
VERSION=0.4
...
AC_PROG_INTLTOOL
ALL_LINGUAS=""
AM_GLIB_GNU_GETTEXT
GETTEXT_PACKAGE=$PACKAGE
AC_SUBST(GETTEXT_PACKAGE)
AC_DEFINE_UNQUOTED([GETTEXT_PACKAGE],
                   ["${GETTEXT_PACKAGE}"], [gettext domain])
...
AC_OUTPUT([
po/Makefile.in
])
...
```

안녕, 글레이드! - gettext (3)

- Modify Makefile.am:

```
SUBDIRS = src pixmaps data po
```

- Modify src/Makefile.am:

```
...
AM_CPPFLAGS = ... -DLOCALE_DIR=\""$(datadir)/locale"\"
...
```

- Create po & po/POTFILES.in:

```
src/main.c
data/hello.glade
```

- Bootstrapping again:

```
$ glib-gettextize -c
$ aclocal;autoheader;touch stamp-h;autoconf;automake -a -c
```

안녕, 글레이드 ! - gettext (4)

- Create & Translate ko.po:

```
$ cd po ; make update-po ; cp hello.pot ko.po
```

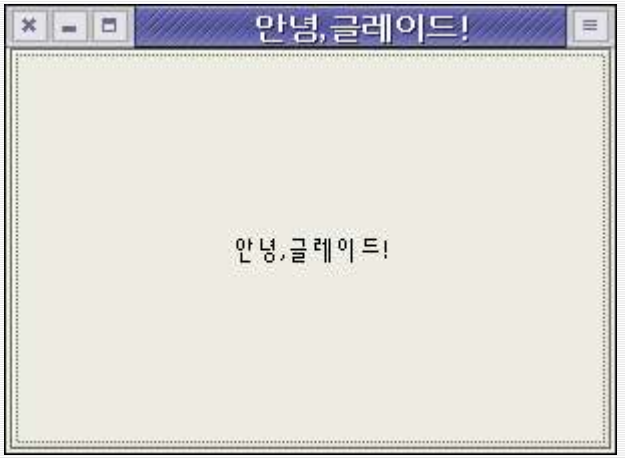
```
...
"Content-Type: text/plain; charset=UTF-8\n"
...
#: src/main.c:15 data/hello.glade:9 data/hello.glade:28
msgid "Hello,Glade!"
msgstr "안녕, 글레이드 !"
```

- Modify configure.in again:

```
...
ALL_LINGUAS="ko"
...
```

- Compile & Run!

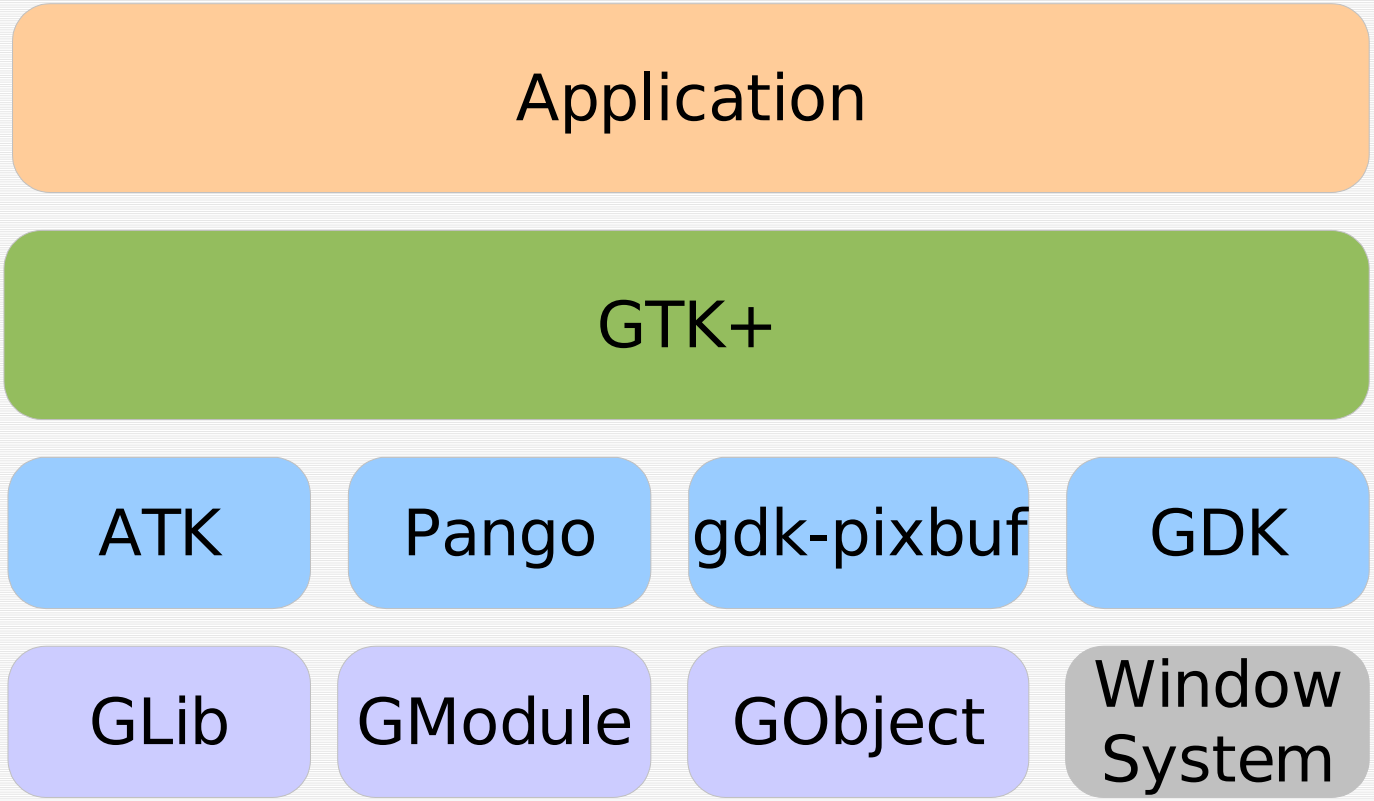
```
$ ./configure ; make ; make install
```



GTK+ Architecture

- Overall Architecture
- GObject Type System
- GObject
- Object Properties
- Ginterface
- Widgets
- Other Features

Overall Architecture (1)



Overall Architecture (2)

- GLib
  - fundamentals: basic types and macros
  - core application support: event loop
  - utilities: thread, string, scanner, xml parser...
  - data types: linked list, hash table, tree ...
  - → *Pleasant, Convenient and Portable*
- GObject
  - generic type system
  - fundamental type implementations
  - signal system & notification mechanism
  - parameter/value system
  - → *Easy to Map (into Other Languages) Object Oriented Framework for C*

Overall Architecture (3)

- GModule
  - portable method for dynamically loading '**plug-ins**'
- Pango (Παν 語)
  - Greek "**Pan**"(All) + Japanese "**Go**"(Language)
  - the layout and rendering framework of internationalized text
- ATK : **A**ccessability **T**ool**K**it
  - a Set of interfaces for accessibility
  - GAIL is an implementation of ATK for GTK+

Overall Architecture (4)

- GDK : **G**IMP **D**rawing **K**it
  - a wrapper around the low-level functions for accessing the underlying windowing functions(Xlib in the case of the X-Window)
  - easy to port into other windowing systems
- GdkPixbuf
  - image loading with custom loader
  - client-side(in memory) image manipulation
  - replaces imlib

Overall Architecture (5)

- GTK+ : **G**IMP **T**ool**K**it
  - C-based Object Oriented API
    - works on many Unix-like platforms(X-Window), Windows, Framebuffer devices, and Console ...
  - Rich Widget Set
    - basic widgets: GtkEntry, GtkButton, ...
    - layout managers: GtkBox, GtkTable
    - complex widgets: TreeView, TextView
    - common dialogs: File/Font/Color/Input
    - action-based Menu/Toolbar
  - Themeable Look & Feel
  - LGPL

GObject Type System

- Object represented by a pointer:

```
GtkWidget *window;
```

- Naming convention for methods:

```
gtk_window_set_title(GTK_WINDOW(window), ...);
gtk_container_add(GTK_CONTAINER(window), ...);
gtk_widget_show_all(window);
```

- Cast macros:

```
GTK_WINDOW(window);
```

- Reference counting - in base GObject class

```
g_object_ref(object);
g_object_unref(object);
```

- SeeAlso GnomeKorea:GObjectTypeSystem

GObject (1)

- Declarations(osd.h)

```
G_BEGIN_DECLS
#define TYPE_OSD              (osd_get_type())
#define OSD(object)           (G_TYPE_CHECK_INSTANCE_CAST((object), \
                                                         TYPE_OSD, Osd))
#define OSD_CLASS(klass)      (G_TYPE_CHECK_CLASS_CAST((klass), \
                                                       TYPE_OSD, OsdClass))
#define IS_OSD(object)        (G_TYPE_CHECK_INSTANCE_TYPE((object), TYPE_OSD))
#define IS_OSD_CLASS(klass)   (G_TYPE_CHECK_CLASS_TYPE((klass), TYPE_OSD))

typedef struct _Osd Osd;
typedef struct _OsdClass OsdClass;

GType osd_get_type(void) G_GNUC_CONST;
Osd *osd_new(void);
...
void osd_set_position(Osd * self, gint x, gint y);
void osd_set_text(Osd * self, const gchar * text);
...
G_END_DECLS
```

GObject (2)

- Definitions (osd.c)

```
struct _Osd {
    GtkWindow parent;
    GtkImage *image;
    ...
};

struct _OsdClass {
    GtkWindowClass parent_class;
    ...
};

Osd *osd_new(void)
{
    Osd *self;
    self = g_object_new(TYPE_OSD, NULL);
    ...
    return self;
}
...
```

GObject (3)

- Type Registeration(osd.c):

```
static void osd_class_init(OsdClass * klass);
static void osd_instance_init(Osd * self);
static void osd_instance_finalize(GObject * gobject);

GType osd_get_type(void)
{
    static const GTypeInfo self_info = {
      sizeof(OsdClass),
      NULL /* base init */, NULL /* base finalize */,
      (GClassInitFunc) osd_class_init,
      NULL /* class finalize */, NULL /* class data */,
      sizeof(Osd), 0,
      (GInstanceInitFunc) osd_instance_init, 0
    };
    static GType self_type = 0;
    if (self_type) return self_type;
    self_type = g_type_register_static(GTK_TYPE_WINDOW, "Osd", &self_info,0);
    return self_type;
}
```

GObject (4)

- Constructor/Destructor(osd.c):

```
static void osd_class_init(OsdClass * klass)
{
    GObjectClass *gobject_class = G_OBJECT_CLASS(klass);
    gobject_class->finalize = osd_instance_finalize;
    ...
}

static void osd_instance_init(Osd * self)
{
    self->image = NULL;
    ...
}

static void osd_instance_finalize(GObject * gobject)
{
    Osd *self = OSD(gobject);
    g_object_unref(self->mask);
    ...
}
```

Object Properties

- Properties are pre-defined attributes of obj ect
- Set:

```
g_object_set (entry,
              "text", "Hello World",
              "cursor_position", 3,
              NULL /* end of parameters */);
```

Property name

Property Value

Terminating NULL

- Get:

```
g_object_get (entry
              "text", &text,
              "cursor_position", &cursor_position,
              NULL);
```

Location to store value

- Emit “notify” signal when it has been chan ged

GInterface

- Sometimes inheritance not enough
  - Need muliple inheritance
- e.g:
  - GtkFileChooserDialog
    - Inherits(extends) from GtkDialog
    - implmenents GtkFileChooser
  - GtkListStore
    - implements GtkTreeModel, GtkTreeSortable, GtkTree DragSource, and GtkTreeDragDest
- Cast macros in same way as inheritance:

```
GtkFileChooser *chooser = GTK_FILE_CHOOSER(dialog);
...
GtkListStore *store = gtk_list_store_new(...)
gtk_tree_view_set_model(view, GTK_TREE_MODEL(store));
```

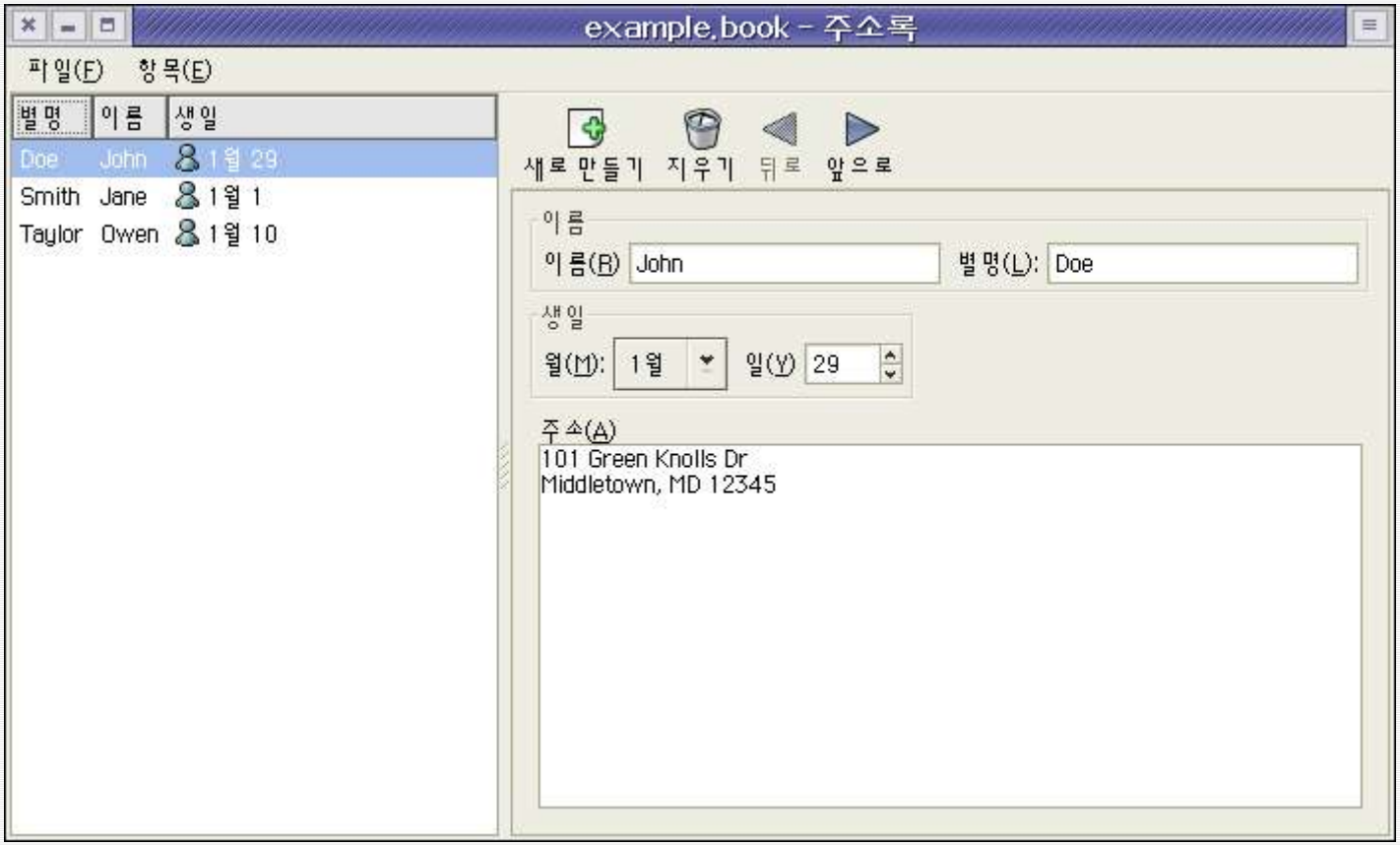
Widgets

- Creating Widget:
  1. gtk_*_new_xxx()
  2. Connect callbacks to signals
  3. Set the attributes of the widget
  4. Pack the widget into a container
  5. gtk_widget_show()
- Casting (SeeAlso GTK+ Widget Hierarchy)
  - G_OBJECT(object)
  - G_CALLBACK(function)
  - GTK_WIDGET (widget)
  - SeeAlso GTK+ Widget Hierarchy
- Each an Widget has its own methods, properties and signals

GTK+ by Example

- What We Make - AddressBook
- Organizing a Program
- Using Object Data
- UI Layout – GtkBox & GtkTable
- GtkDialog
- Model-View?
- GtkTreeView
- Using Boxed Type
- GdkPixbuf

What We Make - AddressBook



Organizing a Program

- Base around application data structures

```
struct AddressBook
{
  char *filename;
  GtkWidget *window; /* top level window */
  GtkWidget *treeview;
  ...
  GtkTextBuffer *address_textbuffer;
  GtkListStore *list_store;
  gboolean in_change;
  gboolean modified;
};
```

- Often have one structure for a top level win dow

Using Object Data

- Way of attaching application data to a widget
- String key identifies each piece of data
  - Store data:

```
g_object_set_data(G_OBJECT(window), "address-book", address_book);
```

  - Get data:

```
address_book = g_object_get_data(G_OBJECT(window), "address-book");
```

- Storing object data on top-level widget avoids having to store it on each individual widget:

```
AddressBook *get_address_book(GtkWidget *widget)
{
   GtkWidget *toplevel;
   toplevel = gtk_widget_get_toplevel(widget);
   return g_object_get_data(toplevel, "address-book");
}
```

UI Layout (1) - GtkBox

- Layout widgets in a row or column:
  - GtkHBox: Horizontally
  - GtkVBox: Vertically
- Box properties:
  - homogeneous – Allocate same space to all cells
  - spacing – Gap between cells
- Cell properties(for each child widget):
  - expand – Allocate remaining(excess) space?(not used with homogeneous)
  - fill – Child fills allocated space?
  - padding – Gap around child(left/right or top/bottom edges)

```
GtkWidget* box = gtk_hbox_new(homogeneous, spacing);
gtk_box_pack_start(GTK_BOX(box), child, expand, fill, padding);
```

UI Layout (2) - GtkTable

- GtkTable allows for grid based layout
- Specify child position by left, right, top, bottom e dges.(Glade specify top, left, colspan, rowspan)
- Also specify options (expand, fill) and padding for each dimension

```
gtk_table_attach(GTK_TABLE(table), child,
                 /* x */                  /* y */
/* attach */     0, 1,                    0, 1,
/* options */    GTK_EXPAND | GTK_FILL, GTK_FILL,
/* padding */    0,                       0);
```

- **Hierarchy of containers defines layout**
- SeeAlso GtkPaned, GtkFixed, GtkButtonBox ...
- SeeAlso **GNOME HIG**

GtkDialog (1)

- Window widget with buttons
  - Each button has a (label, response id) pair
- Two usage of GtkDialog
  - Modal – Easy to use
  - Modaless – IMO, Preferred for the most
- GtkMessageDialog
  - GtkDialog that just holds a message
  - Use printf() style format string

```
gtk_message_dialog_new (parent_window,
                        GTK_DIALOG_DESTROY_WITH_PARENT,
                        GTK_MESSAGE_WARNING,
                        GTK_BUTTONS_OK,
                        "Error when printing: %s",
                        error->message);
```

GtkDialog (2)


```
GtkWidget *dialog;
dialog = gtk_dialog_new_with_buttons("My dialog", main_app_window,
                  GTK_DIALOG_MODAL | GTK_DIALOG_DESTROY_WITH_PARENT,
                                        GTK_STOCK_OK, GTK_RESPONSE_OK,
                                 GTK_STOCK_CANCEL, GTK_RESPONSE_CANCEL,
                                                                 NULL);
GtkResponseType response = gtk_dialog_run(GTK_DIALOG(dialog));
switch (result) {
case GTK_RESPONSE_OK:
      ...
}
gtk_widget_destroy(dialog);
```

pairs of button (label, response id)

```
...
g_signal_connect(dialog, "response", G_CALLBACK(response_cb), NULL);
gtk_widget_show_all(dialog);
...
void response_cb(GtkDialog *dialog, gint response, gpointer data) {
...
}
```

Model-View?

- Widgets so far are small and simple
- GTK+ has two widgets for display of large amounts of information:
  - GtkTextView: multiple line text display
  - GtkTreeView: trees and lists (a list is just a flat tree)
- Split apart widget(**view**) from data store (**model**)
  - GtkTextView: GtkTextBuffer
  - GtkTreeView: GtkTreeModel (GtkListStore, GtkTreeStore, ...)
- Can have multiple views of same model

GtkTextView (1)


```
GtkWidget *view;
GtkTextBuffer *buffer;
GtkTextIter start, end;
char *text;

view = gtk_text_view_new (NULL); /* NULL - create new buffer */
buffer = gtk_text_view_get_buffer (GTK_TEXT_VIEW (view));

/* Set text */
gtk_text_buffer_set_text (buffer, "Some Text");

/* Get text */
gtk_text_buffer_get_start_iter (buffer, &start);
gtk_text_buffer_get_end_iter (buffer, &end);
text = gtk_text_buffer_get_text (buffer,
                                 &start, &end, /* range */
                                 FALSE); /* include invisible? */
g_print ("The text is %s\n", text);
g_free (text);
```

GtkTextView (2)

- Iterator - GtkTextIter
  - Refer to a track of a place in data(GtkTextBuffer)
  - Methods to navigate (iterate) through the data

```
GtkTextIter iter;
gtk_text_buffer_get_start_iter(buffer, &iter);
gtk_text_iter_forward_lines(&iter, 10);
gtk_text_iter_forward_chars(&iter, 10);
```

  - Temporary
    - Changes in buffer invalidate iterator
    - Use GtkTextMark for permanent “bookmark”
- Can add styles(GtkTextTag) to data

```
gtk_text_buffer_insert_with_tags(buffer,&iter,text,len,tag,...);
```

- Can add image(GdkPixbuf) to data

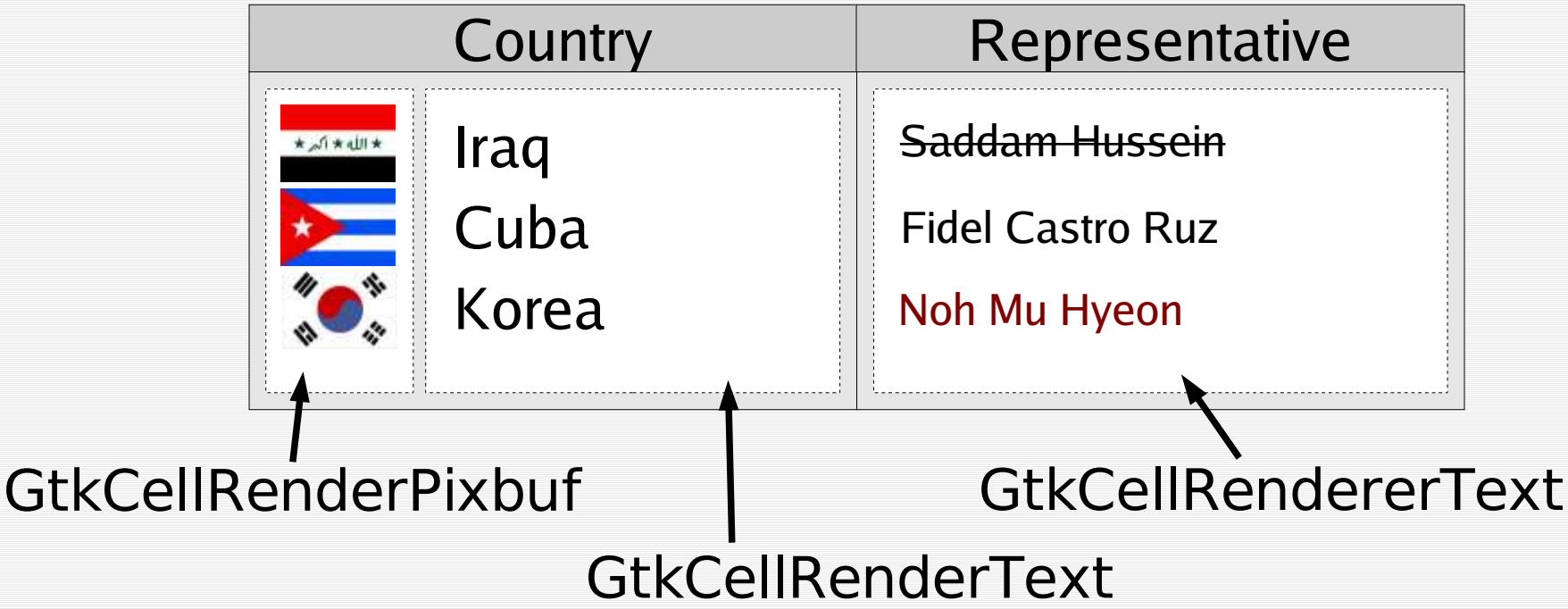
```
gtk_text_buffer_insert_image(buffer, &iter, pixbuf);
```

GtkTreeView (1)

- Handles both lists and trees
- Model/View: Data stored separately from w idget
- GtkTreeView: The widget
- GtkTreeModel: data interface
  - GtkListStore: for Flat data
  - GtkTreeStore: for Heirarchical data
  - Can also create custom models (but difficult) :(

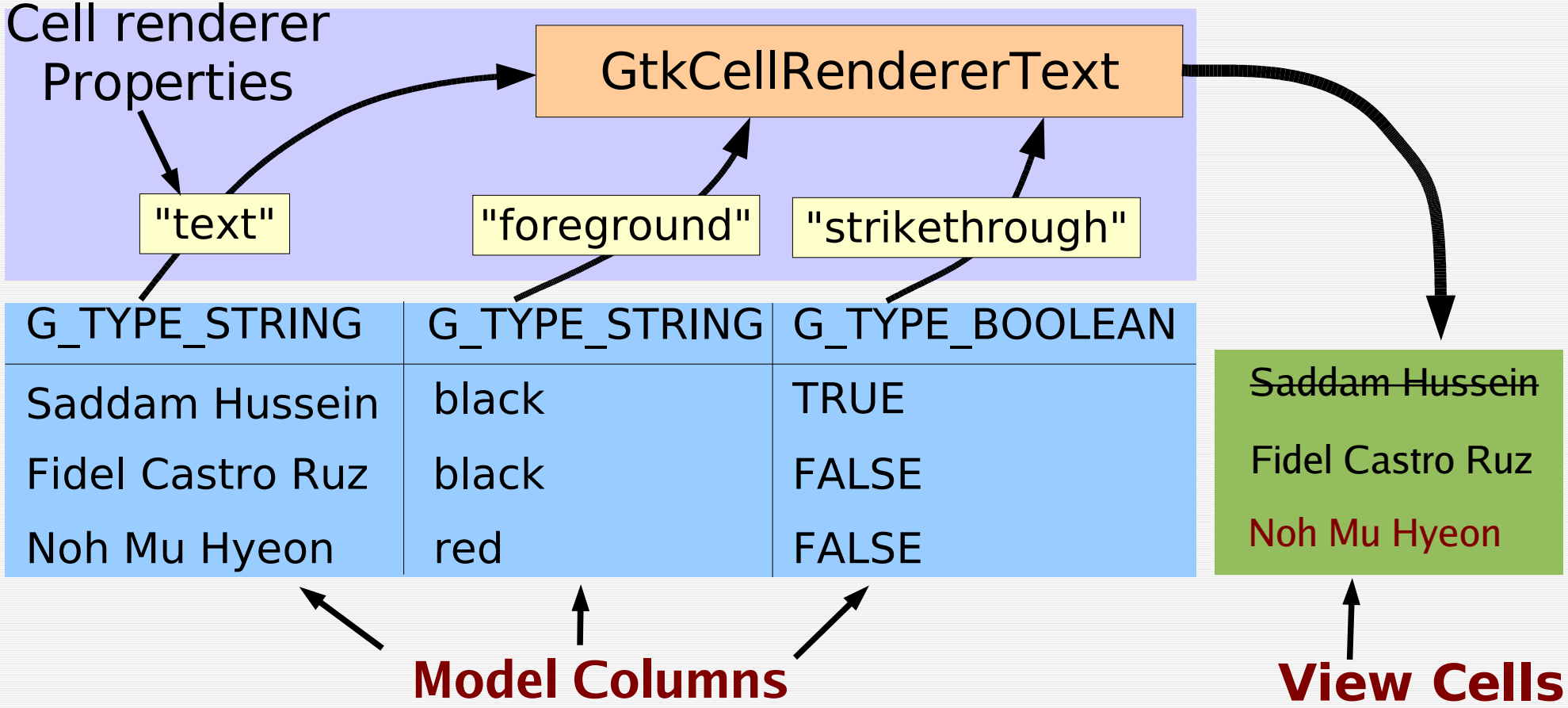
GtkTreeView (2)

- Lists and trees have multiple columns(GtkTreeViewColumn) of information
- Each column drawn by one or more renderers(GtkCellRender*)



GtkTreeView (3)

- Properties for each cell renderer are set, ro w by row, from the GtkTreeModel



GtkTreeView (4)

- Each GtkTreeModel has some number of columns, each with a type
  - G_TYPE_STRING
  - G_TYPE_BOOLEAN
  - G_TYPE_INTEGER
  - G_TYPE_PIXBUF(Image)
  - G_TYPE_POINTER(used with data function)
- Every row in model stores one item in each column
- GtkTreeIter points to a single row in model

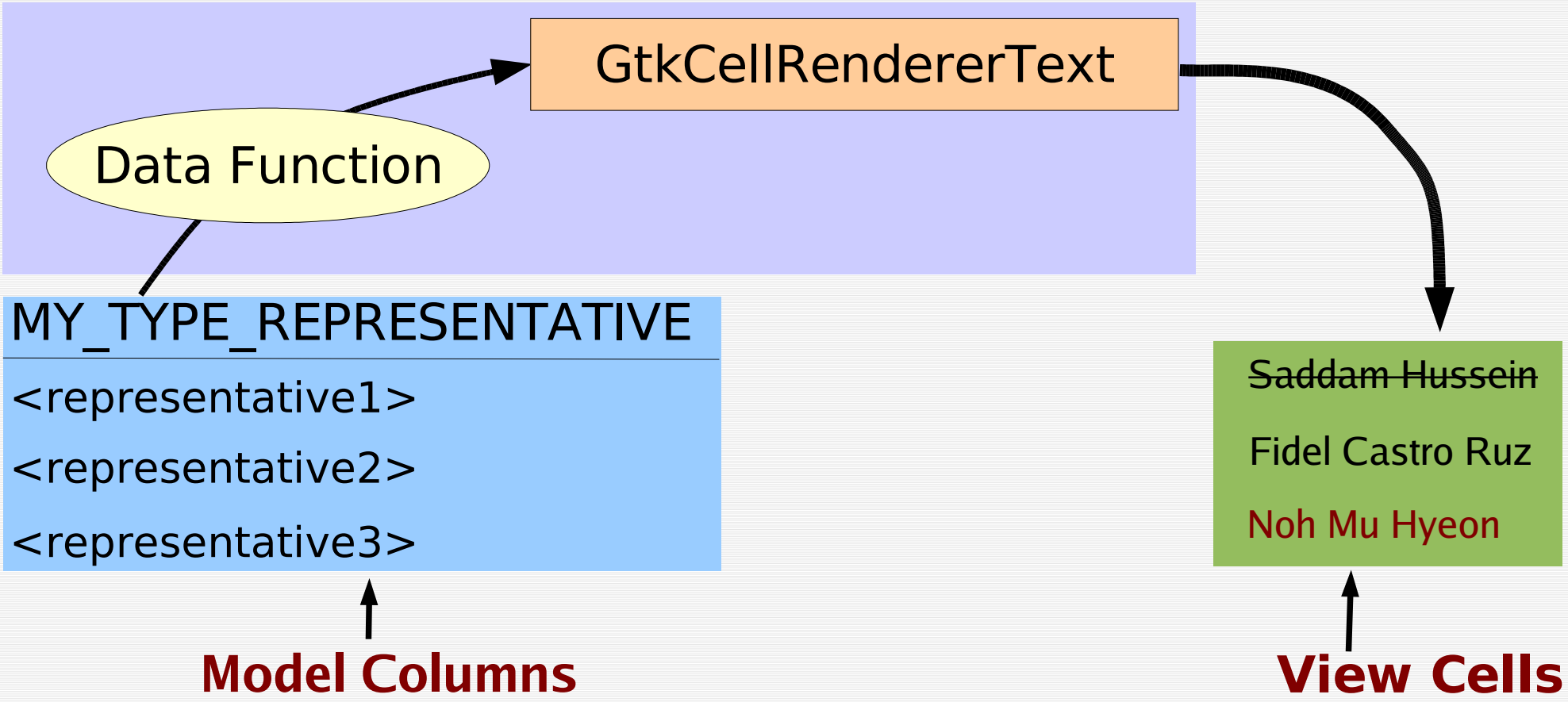
GtkTreeView (5)

- Append a new row:

```
enum { NAME_COLUMN, COLOR_COLUMN, STRIKETHROUGH_COLUMN };

GtkListStore *list_store;
GtkTreeIter iter;

list_store = gtk_list_store_new (3 /* number of columns */,
                                 G_TYPE_STRING,
                                 G_TYPE_STRING,
                                 G_TYPE_BOOLEAN);

gtk_list_store_append (list_store, &iter);

gtk_list_store_set (list_store, &iter,
                    NAME_COLUMN, "Noh Mu Hyeon",
                    COLOR_COLUMN, "red",
                    STRIKETHROUGH_COLUMN, FALSE,
                    -1 /* end of arguments */ );
```

GtkTreeView (6)

- Can use data function to set all properties f rom a single column



GtkTreeView (7)

- Provide cell renderer properties with callback:

```
static void
name_data_func (GtkTreeViewColumn *tree_column,
                GtkCellRenderer   *cell,
                GtkTreeModel      *tree_model,
                GtkTreeIter       *iter,
                gpointer           data)
{
  Representative *rep;

  gtk_tree_model_get (tree_model, iter,
                      REPRESENTATIVE_COLUMN, &rep,
                      -1);
  g_object_set (cell,
                "text", representative_get_name (rep),
                "strikethrough", !representative_made_quota (rep),
                NULL);
  representative_unref (rep);
}
```

Using Boxed Type

- Reference counted structure
- Easy to write than GObject
- e.g. Need to tell GTK+ about AddressEntry in order to use in a GtkListStore

```
#define ADDRESS_TYPE_ENTRY (address_entry_get_type ())

GType address_entry_get_type (void)
{
  static GType our_type = 0;
  if (our_type == 0)
    our_type = g_boxed_type_register_static ("AddressEntry",
                    (GBoxedCopyFunc) address_entry_ref,
                    (GBoxedFreeFunc) address_entry_unref);
  return our_type;
}
```

GtkTreeView (8)

- Selection - GtkTreeSelection
  - Auxilliary object representing current selection
  - Methods for retrieving currently selected rows
  - Signal "selection_changed" for reacting to changes
  - Selection mode setting: gtk_tree_selection_set_mode()
    - GTK_SELECTION_SINGLE: 0 or 1 row selected
    - GTK_SELECTION_BROWSE: 1 row selected
    - GTK_SELECTION_MULTIPLE: 0-N rows selected
- Iteratoration – GtkTreeIter, GtkTreePath, GtkTreeRowReference

GtkTreeView (9)

- Sorting – GtkTreeSortable
  - Idea: User clicks on column header to sort on t hat column
  - But how does GtkTreeView now how to sort on a view column?
    - Might have multiple renderers, custom renderer, etc.
  - Idea: Application sets sort column ID for each c olumn in the view.
  - Default action for sort column ID N is to sort on model column N
  - Possible to also set custom sorting for a sort ID (useful for a data func)

GtkTreeView (10)

- Sorting example:

```
gtk_tree_sortable_set_sort_func (GTK_TREE_SORTABLE (list_store),
                                 SORT_BIRTHDAY,
                                 compare_birthday, NULL, NULL);

int compare_firstname  (GtkTreeModel *model,
                        GtkTreeIter  *a, GtkTreeIter  *b,
                        gpointer      user_data)
{
  AddressEntry *entry_a, *entry_b;
  int result;
  gtk_tree_model_get (model, a, 0, &entry_a, -1);
  gtk_tree_model_get (model, b, 0, &entry_b, -1);
  result = strcmp (entry_a->firstname, entry_b->firstname);
  address_entry_unref (entry_a);
  address_entry_unref (entry_b);
  return result;
}
```

GdkPixbuf (1)

- GdkPixbuf represents an image
- Can load/save many image types
  - PNG, JPEG, GIF, TIFF, BMP, ICO, TGA, ANI, XBM, XPM, WBMP, PCX, RAS
  - Also extensible: librsvg installs SVG loader
  - `$ gdk-pixbuf-query-loaders`
- Direct client-side pixel access.
  - GdkPixbuf - client-side image, directly manipul ate pixels
  - GdkPixmap – server-side image(X11 port), GDK drawing primitives, no direct pixel access

GdkPixbuf (2)

- Scaling & Converting into JPEG:

```
GdkPixbuf *pixbuf, *pixbuf2;
GError *error = NULL;

pixbuf = gdk_pixbuf_new_from_file ("my.png", &error);
if (!pixbuf) {
    g_printerr ("Unable to load image: %s\n",
                error->message);
    g_error_free (error);
} else {
    pixbuf2 = gdk_pixbuf_scale_simple(pixbuf, 100, 100,
                                      GDK_INTERP_BILINEAR);
    gdk_pixbuf_save (pixbuf2, “my.jpg”, "jpeg", &error,
                     "quality", "100", NULL);
    g_object_unref (pixbuf2);
    g_object_unref (pixbuf);
}
```

More Information

- Additonal Libraries
- Language Bindings
- Developer Tools
- References

Additional Libraries

- glib – Portability to Platform(esp. C Runtime)
- gdk – Portability to Window System
- pango - I18n
- atk - A11y
- gdkpixbuf – Image Manipulation
- libgnome/libgnomeui – ...?
- libbonobo/libbonoboui – CORBA based Componen t / Compound Document
- libxml/libxslt – XML parser/XSLT processor
- libgnomedb/libgda – Data access with widgets
- gconf – Registry better than registry
- gnomevfs – Yet another virtual file system
- gstreamer – Multimedia Framework

Language Bindings

Language	Binding	Homepage
Ada	GtkAda	http://libre.act-europe.fr/GtkAda/
C++	Gnomemm/gtkmm	http://www.gtkmm.org/
C++	Inti	http://inti.sourceforge.net/
C#	GTK#/Mono	http://gtk-sharp.sourceforge.net/ http://go-mono.com/
Java	Java-GNOME	http://gtk-sharp.sourceforge.net/
Objective-Caml	LablGTK	http://wwwfun.kurims.kyoto-u.ac.jp/soft/olabl/lablgtk.html
Perl	gtk2-perl	http://gtk2-perl.sourceforge.net/
Python	PyGnome/PyGTK	http://www.daa.com.au/~james/pygtk/
Ruby	Ruby-GNOME2	http://ruby-gnome2.sourceforge.jp/
Scheme	Guile-gtk/Guilt-gobject	http://www.gnu.org/software/guile-gtk/
TCL	Gnocl	http://www.dr-baum.net/gnocl/

Developer Tools

- Anjuta - Full-featured IDE
  - Syntax highlighting editor
  - Debugger
  - Project manager
- Glade - GUI builder
- DevHelp – Fancy document viewer
- MemProf – memory profiler with leak detec tion
- Gob2 - GObject generator
- Meld – Visual diff with CVS support
- vim, emacs and so on...

References

- Owen Tyler's Presentation - http://people.redhat.com/otyler/tutorial/guadec2003/
- Bo Mjewski's Tutorial
- Official GTK 2.0 Tutorial
- API Documents - http://developer.gnome.org/
- GTK+ Official Homepage - http://gtk.org
- GIMP Official Homepage - http://gimp.org
- GNOME Official Homepage - http://gnome.org
- GNOME Korea Official Homepage - http://gnome.or.kr
- IBM developerWorks - http://www.ibm.com/developerworks/
- Special Thanks to http://google.com ;)