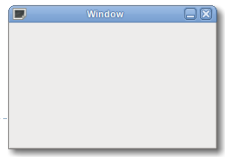
CAD Algorithms for Physical Design - GTK+-based GUI

Christos P Sotiriou

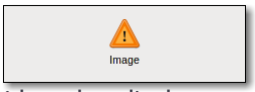
GTK+ Documentation

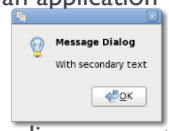
- GTK/GDK (Gimp Tool Kit +) and Cairo
  - http://www.gtk.org/
  - https://developer.gnome.org/gtk2/stable/
  - https://developer.gnome.org/gdk2/stable/
  - https://developer.gnome.org/gtk2/stable/gtk2-General.html
  - http://zetcode.com/gui/gtk2/gtkevents/
  - https://developer.gnome.org/gdk2/stable/gdk2-Cairo-Interaction.html
  - http://cairographics.org/documentation/
  - https://developer.gnome.org/glib/stable/

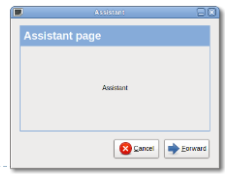
GTK+ Widgets - 1



- Windows
  - GtkDialog — Create popup windows
  - GtkMessageDialog
  - GtkWindow — Toplevel which can contain other widgets
  - GtkAboutDialog — Display information about an application
  - GtkAssistant
- Display Widgets
  - GtkImage
  - GtkLabel — A widget that displays a small to medium amount of text
  - GtkProgressBar — A widget which indicates progress visually
  - GtkStatusbar — Report messages of minor importance to the user
  - GtkInfoBar
  - GtkStatusIcon — Display an icon in the system tray
  - GtkSpinner

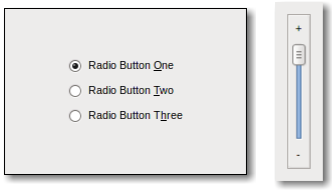




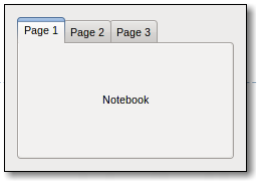


GTK+ Widgets - 2

- Buttons and Toggles
  - GtkButton — A widget that creates a signal when clicked on
  - GtkCheckButton — Create widgets with a discrete toggle button
  - GtkRadioButton — A choice from multiple check buttons
  - GtkToggleButton — Create buttons which retain their state
  - GtkLinkButton — Create buttons bound to a URL
  - GtkScaleButton — A button which pops up a scale
  - GtkVolumeButton



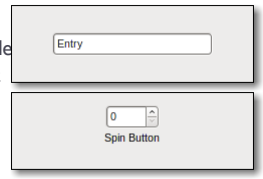
GTK+ Widgets - 3



- Layout Containers
  - GtkHBox
  - GtkVBox
  - GtkHButtonBox — A container for arranging buttons horizontally
  - GtkVButtonBox
  - GtkFixed — A container which allows you to position widgets at fixed coordinates
  - **GtkHPaned** — A container with two panes arranged horizontally
  - **GtkVPaned**
  - GtkLayout — Infinite scrollable area containing child widgets and/or custom drawing
  - GtkNotebook — A tabbed notebook container
  - GtkTable — Pack widgets in regular patterns
  - GtkExpander — A container which can hide its child
  - GtkOrientable — An interface for flippable widgets

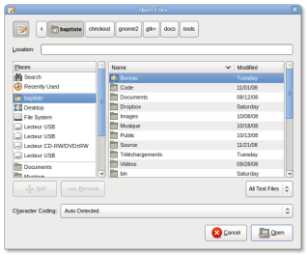
GTK+ Widgets - 4

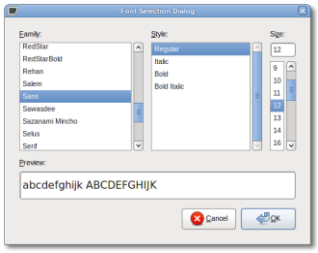
- Action-based menus and toolbars
  - GtkUIManager — Constructing menus and toolbars from an XML description
- Interface builder
  - GtkBuilder — Build an interface from an XML UI definition
- Scrolling
  - **GtkHScrollbar** — A horizontal scrollbar
  - **GtkVScrollbar**
  - GtkAdjustment — A GtkObject representing an adjustable bounde
  - GtkRange — Base class for widgets which visualize an adjustment
  - GtkScrolledWindow — Adds scrollbars to its child widget
- Numeric/Text Data Entry
  - GtkEntry — A single line text entry field
  - GtkEntryBuffer — Text buffer for GtkEntry
  - GtkEntryCompletion — Completion functionality for GtkEntry
  - GtkHScale — A horizontal slider widget for selecting a value from a range
  - GtkVScale
  - GtkSpinButton — Retrieve an integer or floating-point number from the user
  - GtkEditable — Interface for text-editing widgets



GTK+ Widgets - 5

- Selectors (File/Font/Color/Input Devices)
  - GtkFileChooserDialog — A file chooser dialog, suitable for "File/Open" or "File/Save" commands
  - GtkFileChooserWidget — File chooser widget that can be embedded in other widgets
  - GtkFileFilter — A filter for selecting a file subset
  - GtkFontSelectionDialog — A dialog box for selecting fonts





GDK Cursors

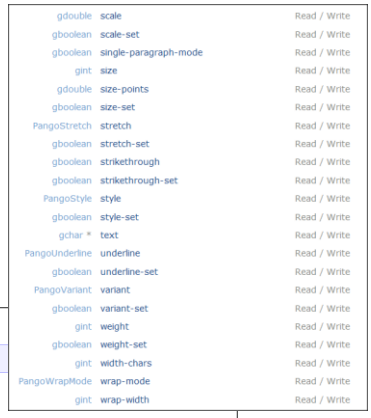
- https://developer.gnome.org/gdk3/stable/gdk3-Cursors.html

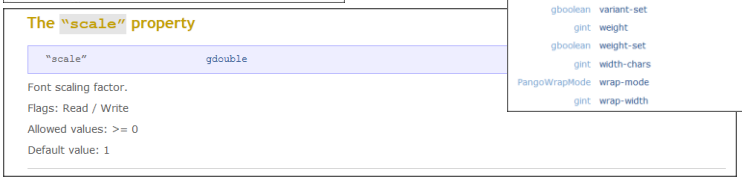
Creates a new cursor from the set of builtin cursors. Some useful ones are:

- GDK_RIGHT_PTR (right-facing arrow)
- GDK_CROSSHAIR (crosshair)
- GDK_XTERM (I-beam)
- GDK_WATCH (busy)
- GDK_FLEUR (for moving objects)
- GDK_HAND1 (a right-pointing hand)
- GDK_HAND2 (a left-pointing hand)
- GDK_LEFT_SIDE (resize left side)
- GDK_RIGHT_SIDE (resize right side)
- GDK_TOP_LEFT_CORNER (resize northwest corner)
- GDK_TOP_RIGHT_CORNER (resize northeast corner)
- GDK_BOTTOM_LEFT_CORNER (resize southwest corner)
- GDK_BOTTOM_RIGHT_CORNER (resize southeast corner)
- GDK_TOP_SIDE (resize top side)
- GDK_BOTTOM_SIDE (resize bottom side)
- GDK_SB_H_DOUBLE_ARROW (move vertical splitter)
- GDK_SB_V_DOUBLE_ARROW (move horizontal splitter)

Widget Properties - 1

- All widgets possess properties:





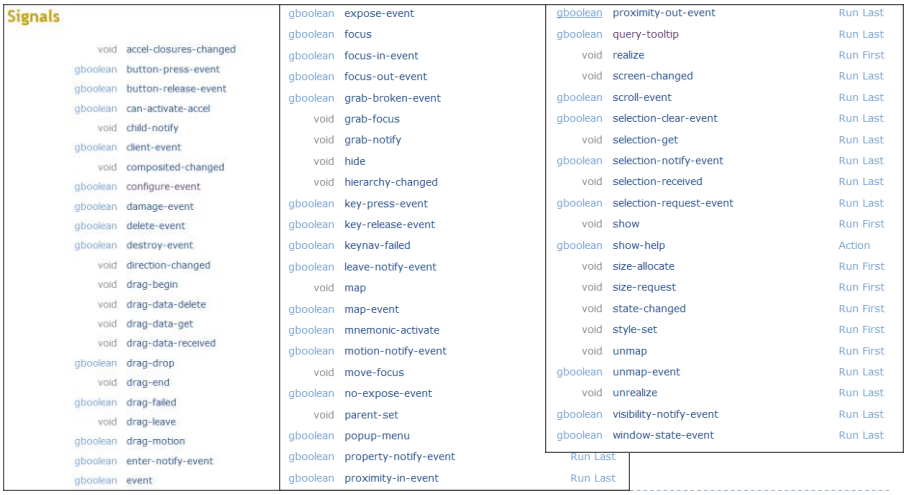
Widget Properties - 2

- Modify

- `// se`
  `fonts`
  `hiera`
  `g_obj`
  `ale the`
  `NULL);`

Widget Signals - 1

- All widgets possess signals:



Signals

void	accel-closures-changed
gboolean	button-press-event
gboolean	button-release-event
gboolean	can-activate-accel
void	child-notify
gboolean	client-event
void	composited-changed
gboolean	configure-event
gboolean	damage-event
gboolean	delete-event
gboolean	destroy-event
void	direction-changed
void	drag-begin
void	drag-data-delete
void	drag-data-get
void	drag-data-received
gboolean	drag-drop
void	drag-end
gboolean	drag-failed
void	drag-leave
gboolean	drag-motion
gboolean	enter-notify-event
gboolean	event
gboolean	expose-event
gboolean	focus
gboolean	focus-in-event
gboolean	focus-out-event
gboolean	grab-broken-event
void	grab-focus
void	grab-notify
void	hide
void	hierarchy-changed
gboolean	key-press-event
gboolean	key-release-event
gboolean	keynav-failed
gboolean	leave-notify-event
void	map
gboolean	map-event
gboolean	mnemonic-activate
gboolean	motion-notify-event
void	move-focus
gboolean	no-expose-event
void	parent-set
gboolean	popup-menu
gboolean	property-notify-event
gboolean	proximity-in-event
gboolean	proximity-out-event
gboolean	query-tooltip
void	realize
void	screen-changed
gboolean	scroll-event
gboolean	selection-clear-event
void	selection-get
gboolean	selection-notify-event
void	selection-received
gboolean	selection-request-event
void	show
gboolean	show-help
void	size-allocate
void	size-request
void	state-changed
void	style-set
void	unmap
gboolean	unmap-event
void	unrealize
gboolean	visibility-notify-event
gboolean	window-state-event

Run Last	
Run Last	
Run First	
Run Last	
Run Last	
Run Last	
Run Last	
Run Last	
Run Last	
Run First	
Action	
Run First	
Run First	
Run First	
Run First	
Run First	
Run Last	
Run Last	
Run Last	
Run Last	
Run Last	

Widget Signals - 2

- https://developer.gnome.org/gobject/unstable/gobject-Signals.html#g-signal-connect

g_signal_connect()

```
#define             g_signal_connect(instance, detailed_signal, c_handler, data)
```

Connects a GCallback function to a signal for a particular object.

The handler will be called before the default handler of the signal.

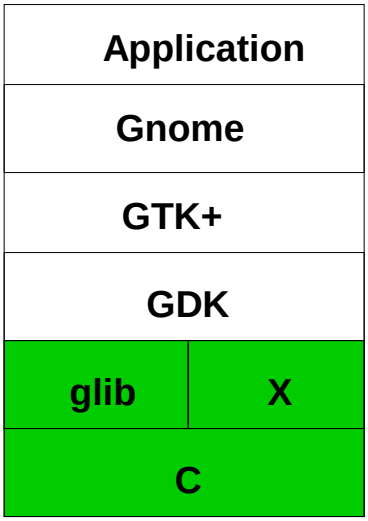
See memory management of signal handlers for details on how to handle the return value and memory management of `data` .

Parameters

instance	the instance to connect to.
detailed_signal	a string of the form "signal-name::detail".
c_handler	the GCallback to connect.
data	data to pass to `c_handler` calls.

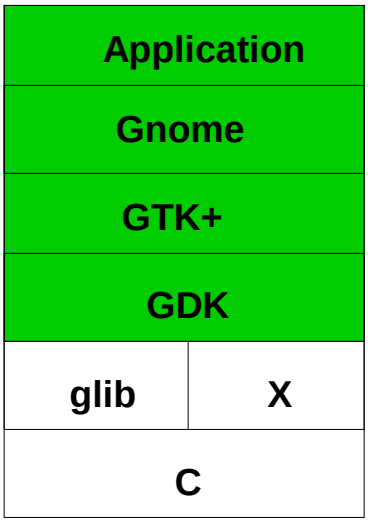
Returns

the handler id (always greater than 0 for successful connections)



- X11 Graphics Library : *low-level functions to control the display*
- glib : *library of C functions, macros and structs used by GDK, GTK+ and GNOME*
- C Standard Libraries
- Linux System Calls

Figure 1.1: The levels of software for a GNOME application in Linux



- Gnome application program : *you will write this*
- GNOME : *extension of GTK+, specialized widgets*
- GIMP Toolkit: *organizes the GDK functions into objects providing the functionality of widgets*
- GIMP Drawing Kit : *simplifies access to X functions*

Figure 1.1: The levels of software for a GNOME application in Linux

OOP in C

Object oriented programming (OOP) is a method of organizing the data and code.

The GTK and GNOME libraries are written in C.

Some languages (C++, Java) automate object oriented programming and provide language features to enforce the OO data access protocols.

GTK+ and GNOME libraries are written in C, thus the object orientation must be simulated.

Terminology

Encapsulation - *object encapsulates its data. Data can only be accessed through function calls designed for that purpose.*

Inheritance - *properties that are available through a base class upon which a new class is based.*

Methods - *the above mentioned functions also used to create a new object from a class.*

Class - definition code used to construct an object

Object - a particular instance of a class

Terminology

Each of the "classes" in GTK+ are actually created using **`structs`**. These **`structs`** contain data fields which serve the purpose that private data members would in C++. They are not protected from improper access.

"Methods" for each class of object are simple functions, whose first parameter is the type of object on which the function is defined.

Inheritance is achieved by including the entire struct of a base class.

Terminology

Event - *sent to your program when a hardware action such as a keystroke, movement of the mouse, pressing of a mouse button, takes place or when X window changes occur in the display (window appears, window is uncovered, etc)*

Signal - *Events are translated into signals. They are sent to your program when one or several hardware actions take place or when certain things occur within a widget (scroll bar moves, button is pressed)*

Callback function - *a function which is called by your program upon receipt of a particular signal.*

Example p.24

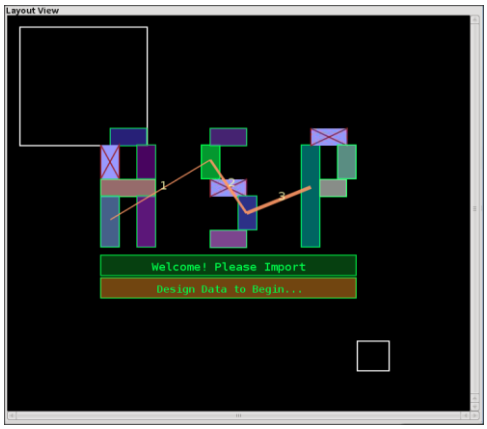
```
/** gtkwin.c **/
#include <gtk/gtk.h>

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

     gtk_init(&argc,&argv);
     topLevelWindow =
          gtk_window_new(GTK_WINDOW_TOPLEVEL);
     gtk_widget_show(topLevelWindow);
     gtk_main();
     exit(0);
}
```


makefile

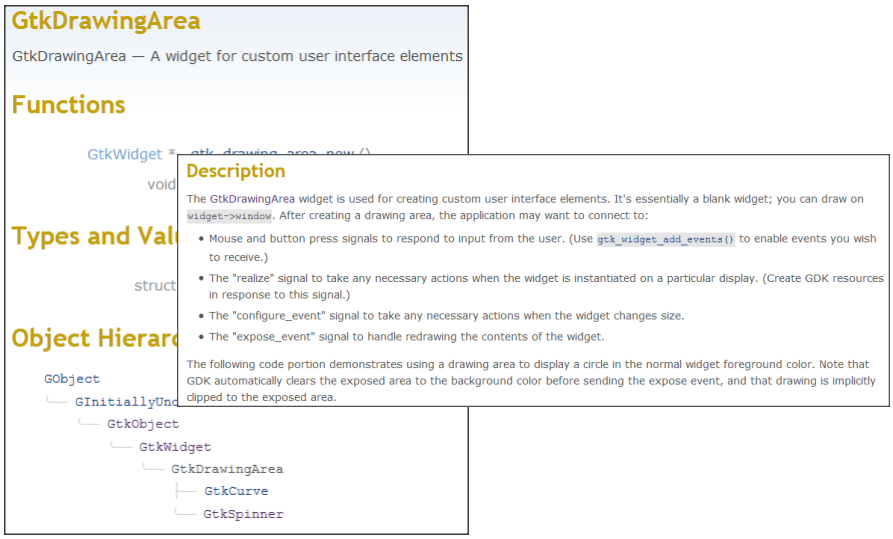
```
CC=gcc
LDLIBS=`gtk-config --libs`
CFLAGS=-Wall -g `gtk-config --cflags`

gtkwin: gtkwin.o
     $(CC) $(LDLIBS) gtkwin.o -o gtkwin

gtkwin.o: gtkwin.c
     $(CC) $(CFLAGS) -c gtkwin.c

clean:
     rm -f gtkwin
     rm -f *.o
```

Layout Canvas with Scrollbars

- Layout View Demo



GtkDrawingArea Widget



Layout Canvas with Scrollbars Example

Layout Canvas with Scrollbars Example – page 1

```
int main()
{
  GtkWidget *maincanvashbox; // contains maincanvas and vertical scrollbar //
  GtkWidget *maincanvasvbox; // contains maincanvashbox and horizontal scrollbar //
  GtkWidget *mainframe; // main canvas frame //


  gtk_init(NULL,  NULL);

  aspwindow = gtk_window_new(GTK_WINDOW_TOPLEVEL);

  gtk_window_set_title(GTK_WINDOW(aspwindow), "ASP");
  gtk_window_set_default_size(GTK_WINDOW(aspwindow), 300, 300); // default size //
  g_signal_connect(G_OBJECT(aspwindow), "destroy", G_CALLBACK(quitaction), aspwindow);

  maincanvashbox = gtk_hbox_new(FALSE, 0);
  maincanvasvbox = gtk_vbox_new(FALSE, 0);

  mainframe = gtk_frame_new("Layout View");
  maincanvas = gtk_drawing_area_new();
  gtk_widget_set_size_request(maincanvas, maincanvasWidth, maincanvasHeight);

// maincanvas Event Handlers //
  g_signal_connect(G_OBJECT(maincanvas), "expose-event", G_CALLBACK(expose), maincanvas);

  gtk_widget_add_events(maincanvas, GDK_SCROLL);
  gtk_widget_add_events(maincanvas, GDK_BUTTON_PRESS_MASK);

  g_signal_connect(G_OBJECT(maincanvas), "scroll-event", G_CALLBACK(scroll), maincanvas);
  g_signal_connect(G_OBJECT(maincanvas), "button-press-event", G_CALLBACK(mousebutton), maincanvas);

  maincanvasvscrollbaradjustment = gtk_adjustment_new(0.0, 0.0, 0.0, 0.0, 0.0, 0.0);
  maincanvashscrollbaradjustment = gtk_adjustment_new(0.0, 0.0, 0.0, 0.0, 0.0, 0.0); . . .
```


Layout Canvas with Scrollbars Example

Layout Canvas with Scrollbars Example – page 2

```
. . .

  maincanvasvscrollbar = gtk_vscrollbar_new(GTK_ADJUSTMENT(maincanvasvscrollbaradjustment));
  maincanvashscrollbar = gtk_hscrollbar_new(GTK_ADJUSTMENT(maincanvashscrollbaradjustment));

  g_signal_connect(G_OBJECT(maincanvasvscrollbar), "change-value", G_CALLBACK(maincanvasvscroll),
maincanvasvscrollbar);
  g_signal_connect(G_OBJECT(maincanvashscrollbar), "change-value", G_CALLBACK(maincanvashscroll),
maincanvashscrollbar);

  gtk_box_pack_start(GTK_BOX(maincanvashbox), maincanvas, FALSE, FALSE, 0);
  gtk_box_pack_start(GTK_BOX(maincanvashbox), maincanvasvscrollbar, FALSE, FALSE, 0);

  gtk_box_set_child_packing(GTK_BOX(maincanvashbox), maincanvas, TRUE, TRUE, 0, GTK_PACK_START);
  gtk_box_set_child_packing(GTK_BOX(maincanvashbox), maincanvasvscrollbar, FALSE, FALSE, 0, GTK_PACK_END);

  gtk_box_pack_start(GTK_BOX(maincanvasvbox), maincanvashbox, FALSE, FALSE, 0);
  gtk_box_pack_start(GTK_BOX(maincanvasvbox), maincanvashscrollbar, FALSE, FALSE, 0);

  gtk_box_set_child_packing(GTK_BOX(maincanvasvbox), maincanvashbox, TRUE, TRUE, 0, GTK_PACK_START);
  gtk_box_set_child_packing(GTK_BOX(maincanvasvbox), maincanvashscrollbar, FALSE, FALSE, 0, GTK_PACK_END);

  gtk_container_add(GTK_CONTAINER(mainframe), maincanvasvbox);

  g_signal_connect(G_OBJECT(maincanvas), "size-allocate", G_CALLBACK(resizemaincanvas), maincanvas);

  gtk_container_add(GTK_CONTAINER(aspwindow), mainframe);

  gtk_widget_show_all(aspwindow);

  gtk_main();

}
```

GTK to Cairo Interaction

- http://cairographics.org/documentation/
- http://cairographics.org/manual/
- http://cairographics.org/tutorial/

```
// Expose-Event Paint Function for maincanvas //
static void maincanvaspaint(GtkWidget *widget, GdkEventExpose *event, gpointer data)
{
  maincanvas_drawable = maincanvas->window; // drawable drawing area window //

  maincanvas_cs = gdk_cairo_create(maincanvas_drawable); // corresponding cairo state //

  // draw  ... in Cairo //

  cairo_destroy(maincanvas_cs);
}
```

cairo

Cairo Drawing Functions in a Nutshell

- cairo_move_to(cairo_t *cr, double x, double y);
- cairo_line_to(cairo_t *cr, double x, double y);
- cairo_rectangle(cairo_t *cr, double x, double y, double width, double height);
- cairo_rel_line_to(cairo_t *cr, double x, double y);
- cairo_rel_move_to(cairo_t *cr, double x, double y);
- cairo_select_font_face(cairo_t *cr, const char *family, cairo_font_slant_t slant, cairo_font_weight_t weight);
- cairo_set_font_size(cairo_t *cr, doble size);
- cairo_show_text(cairo_t *cr, const char *utf8);
- cairo_set_source_rgb(cairo_t *cr, double red, double green, double blue);
- cairo_set_source_rgba(cairo_t *cr, double red, double green, double blue, double alpha);