Linux GUI development

GNOME GDK 2.0 development

- Heather Lomond

Introduction

- **There are many GUI development environments.**
  - **QT is the main big contender to Gnome**
  - **Gnome Developer tools are just one example**
- **This is a guide to using GDK 2.0.**
- **It is out of date now (GDK 3.0 has replaced it)**
- **But:**
  - **it is quite a nice simple way to do things**
  - **it has quite a small footprint compared to others**
  - **It is available as a package for most things**

Setting it up on your distro

This can be quite tricky, but, if you are lucky all you will need is:

```
apt-get install libgtk2.0-dev
```

Includes

We need some include files to help out. This does it all

```
#include <gtk/gtk.h>
```

Makefile

This is the complex bit. GTK uses pkg-config to set up the library paths etc:

```
CFLAGS += -Wall -Wextra -Wunused -DVERSION=\"${VER}\" `pkg-config --cflags gtk+-2.0` `pkg-config --libs gtk+-2.0` -I../../code/hal/ -I../../code/soc/ -I../../code/common/ -lm

$(BIN): ${OBJ}
	export PKG_CONFIG_PATH=/usr/lib/x86_64-linux-gnu/pkgconfig/
	@echo "  LD     "$@
	@${CC} ${CFLAGS} -o $@ ${OBJ} ${LDFLAGS}

%.o: %.c
	@echo "  CC     "$<
	@${CC} ${CFLAGS} -c -fPIC -o $@ $<
```

Global Variables

We need some global variables to help out:

```
/* GUI stuff */
GtkWidget *drawing;
GdkFont *font, *font_small, *font_big;
GdkGC *my_gc;
GdkColor my_green;
GdkColor my_amber;
GdkColor my_red;
GdkColor my_blue;
GdkColor my_grey;
```

Setting up the world (1)

First off we need to create some variables to point to our window and things in it:

```
/* this is where it all starts */
int main( int   argc, char *argv[] )
{
    /* graphics variables */
    GtkWidget *window;
    GtkWidget *box;
```

Then we can initialise the gtk environment:

```
    /* now initialise the display */
    gtk_init (&argc, &argv);
```

Setting up the world (2)

Now, for this application, we want to read some sensors and update the display on a periodic basis:

```
/* create a timed function call to update the display */
g_timeout_add(250, timer_function, 0); /*0.25 second timeout */
```

We also want to use some fonts of different sizes

```
/* setup some fonts for use elsewhere */
font= gdk_font_load("
        -adobe-helvetica-medium-r-normal--*-140-*-*-*-*-*-*");
font_small= gdk_font_load("-DejaVu-Sans--*-100-*-*-*-*-*-*");
font_big= gdk_font_load(
      "-*-helvetica-bold-r-normal-*-35-120-*-*-*-*-iso8859-1");
```

Setting up the world (3)

And now we are ready to actually create our window:

```
/* create the window and look and feel */
window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
gtk_window_set_title (GTK_WINDOW(window),
                                        "Sheep Dog Puppy v1.0");
gtk_container_set_border_width (GTK_CONTAINER (window), 10);
```

We will want to be able to stop our window with the little cross (top right):

```
/* this allows us to stop the program */
g_signal_connect (window, "destroy",
                  G_CALLBACK (destroy), NULL);
```

Setting up the world (4)

Next we want to put something into the window This is where you would put menu bars, draw images, add boxes that you might want to fill in(forms) etc.:

```
/* put something in the window to draw onto */
box = gtk_vbox_new (FALSE, 0);
gtk_container_add (GTK_CONTAINER (window), box);
drawing = gtk_drawing_area_new();
gtk_drawing_area_size (GTK_DRAWING_AREA(drawing), 1430, 722);
```

And it will need a routine that tells it how to redraw the window if someone obscures part of it:

```
/* tell it how to be undated by the GUI manager */
g_signal_connect(G_OBJECT(drawing), "expose-event",
                                 G_CALLBACK(expose_event), NULL);
```

Setting up the world (5)

It is also useful to set up some colour definitions:

```
/* set up some colours for use in other places */
my_red.red =   60000;
my_red.green = 30000;
my_red.blue =  30000;
my_amber.red =   60000;
my_amber.green = 50000;
my_amber.blue =      0;
my_green.red =       0;
my_green.green = 60000;
my_green.blue =      0;
my_blue.red =   0;
my_blue.green = 0;
my_blue.blue =  60000;
my_grey.red =   40000;
my_grey.green = 40000;
my_grey.blue =  40000;
```

Setting up the world (6)

Once set up, we can assemble it all together:

```
/* put it all together and display it */
gtk_box_pack_start (GTK_BOX(box), drawing, TRUE, TRUE, 0);
gtk_widget_show(drawing);
gtk_widget_show(box);
```

And we can actually display it with this:

```
/* Finish up and start the window manager */
gtk_widget_show (window);
```

Setting up the world (7)

The last step is to transfer control to GTK to keep track of events (mouse clicks timers etc). Note this doesn't return!

```
    gtk_main ();

     return 0;
}
```

Drawing text

Draw some text with:

```
gdk_draw_text(drawing->window,font_big,
              drawing->style->white_gc,
              INC_X+250, INC_Y+10*tick+9, text, 3);
```

(note the last 3 is the number of characters to draw)

Colours

GTK uses graphics contexts (gc) to tell it how to display stuff. These contain colour info.

```
my_gc=drawing->style->fg_gc[GTK_STATE_NORMAL];
gdk_gc_set_rgb_fg_color(my_gc, &my_blue);
```

And we can use this with:

```
gdk_draw_line(drawing->window,   my_gc,   X, Y,  X+100, Y+90);
```

Some GCs are predefined:

```
drawing->style->white_gc
drawing->style->black_gc
```

Fonts

Fonts on most systems can be found in the file:

```
/etc/X11/fonts/misc/xfonts-base.alias
```

You can use wildcards in the name you use, but you cannot do this for all parameters (so if you define wildcard the X then you have to define the Y size etc.).

```
-adobe-helvetica-medium-r-normal--*-140-*-*-*-*-*-*
```

The ‘140’ in this definition is the 10 times the Point Size that this font was designed for (in this case 14pt).

Timer

Remember we set up a timer? Here is what the call looks like:

```
/* this routine is called periodically by the GUI it does all the
real work */
gint timer_function(gpointer data)
{
.
.
.
    I use this to read all the sensors and display their numbers on
    the screen
}
```

The expose_event routine

This callback was set up in main. It tells the window manager how to draw the window in case it is obscured

```
/* this is called then we initially start up and when the image is
blocked  and then uncovered etc */
gboolean expose_event(GtkWidget *drawing, GdkEventExpose *event,
gpointer data)
{
.
.
.
     This draws all the background text, box outlines etc.
}
```

Destroy callback

We need to set this up to enable the program to end

```
/* this is called to end the GUI and finish the program */
static void destroy()
{
    gtk_main_quit();
}
```

Drawing Lines and Rectangles

Draw a line with:

```
gdk_draw_line(drawing->window,drawing->style->white_gc,
                          INC_X+10, INC_Y+10*tick,
                          INC_X+30, INC_Y+10*tick);
```

Draw a rectangle with:

```
gdk_draw_rectangle (drawing->window,
                    drawing->style->black_gc,
                    TRUE, 0, 0,
                    1430, 722);
```

Drawing Circles

Draw a circle with:

```
gdk_draw_arc(drawing->window,drawing->style->black_gc, TRUE,
                    INC_X+102, INC_Y+(90-old_target)*4-18,
                    36, 36, 0, 360*64);
```

Where to get info

Everything about GDK 2.0 can be found online at:

`https://developer.gnome.org/gdk2/stable/`

Questions