```
#include <stdlib.h>

#include "x3.h"

void x3init(int *pargc, char ***pargv)
{
    gtk_init(pargc, pargv);
}

static void
x3_getfirst_callback(GtkWidget *widget, gpointer data)
{
    GtkWidget **pwidget = (GtkWidget **)data;

    if (*pwidget == NULL)
     *pwidget = widget;
}

static void x3widget_init(x3widget *w, x3widget *parent, char *name,
                 GtkWidget *widget)
{
    w->name = g_strdup(name);
    w->widget = widget;
    w->parent = parent;
    if (parent) {
     if (GTK_IS_WINDOW(parent->widget)) {
         GtkWidget *vbox = NULL;

         gtk_container_foreach(GTK_CONTAINER(parent->widget),
                      x3_getfirst_callback,
                      (gpointer)&vbox);

         if (GTK_IS_MENU_ITEM(widget)) {
          GtkWidget *first_child = NULL;
          GtkWidget *menubar;

          gtk_container_foreach(GTK_CONTAINER(vbox),
                       x3_getfirst_callback,
                       (gpointer)&first_child);
          if (first_child == NULL || !GTK_IS_MENU_BAR(first_child)) {
              menubar = gtk_menu_bar_new();
              gtk_box_pack_start(GTK_BOX(vbox), menubar,
                         FALSE, FALSE, 0);
              gtk_widget_show(menubar);
          } else
              menubar = first_child;
          gtk_menu_bar_append(GTK_MENU_BAR(menubar), widget);
         } else {
          gtk_container_add(GTK_CONTAINER(vbox), widget);
         }
     } else if (GTK_IS_MENU_ITEM(parent->widget)) {
         GtkWidget *menu = gtk_menu_item_get_submenu(GTK_MENU_ITEM(parent-
>widget));

         gtk_menu_shell_append(GTK_MENU_SHELL(menu), widget);
     } else {
         gtk_container_add(GTK_CONTAINER(parent->widget), widget);
     }
    }
}

static x3widget *x3widget_new(x3widget *parent, char *name, GtkWidget *widget)
{
    x3widget *result = (x3widget *)malloc(sizeof(x3widget *));

    x3widget_init(result, parent, name, widget);
    return result;
}

void x3_window_show(x3widget *w)
{
    gtk_widget_show(w->widget);
}

void x3main(void)
{
    gtk_main();
}

/* some constructors */

typedef struct {
    x3widget base;
    x3window_callback callback;
    void *callback_data;
} x3widget_window;

x3widget *x3window(x3windowflags flags, char *label,
             x3window_callback callback, void *callback_data)
{
    GtkWidget *window;
    GtkWidget *vbox;
    x3widget_window *result;

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
    gtk_window_set_title(GTK_WINDOW(window), label);
    vbox = gtk_vbox_new(FALSE, 0);
    gtk_widget_show(vbox);
    gtk_container_add(GTK_CONTAINER(window), vbox);
    result = (x3widget_window *)malloc(sizeof(x3widget_window));
    x3widget_init(&result->base, NULL, "mainwin", window);
    result->callback = callback;
    result->callback_data = callback_data;
    return &result->base;
}

x3widget *x3menu(x3widget *parent, char *name)
{
    GtkWidget *item;
    GtkWidget *menu;

    menu = gtk_menu_new();

    item = gtk_menu_item_new_with_label(name);
    gtk_widget_show(item);

    gtk_menu_item_set_submenu(GTK_MENU_ITEM(item), menu);
    return x3widget_new(parent, NULL, item);
}

typedef struct {
    x3widget base;
    char *cmd;
} x3widget_cmdable;

static void x3cmdable_clicked(GtkWidget *widget, gpointer data)
{
    x3widget_cmdable *wc = (x3widget_cmdable *)data;
    char *cmd = wc->cmd;
    x3widget *w = &wc->base;
    x3widget_window *ww;

    while (w->parent) w = w->parent;
    ww = (x3widget_window *)w;
    ww->callback(w, ww->callback_data, cmd, "command", NULL, NULL);
}

x3widget *x3menuitem(x3widget *parent, char *name, char *cmd, char *shortcut)
{
    GtkWidget *item;
    x3widget_cmdable *result = (x3widget_cmdable *)malloc(sizeof(x3widget));


    item = gtk_menu_item_new_with_label(name);
    x3widget_init(&result->base, parent, cmd, item);
    result->cmd = g_strdup(cmd);
    g_signal_connect(G_OBJECT(item), "activate",
                     G_CALLBACK(x3cmdable_clicked), result);
    gtk_widget_show(item);
    return &result->base;
```

```
}

x3widget *x3menusep(x3widget *parent)
{
    GtkWidget *item;

    item = gtk_separator_menu_item_new();
    gtk_widget_show(item);
    return x3widget_new(parent, NULL, item);
}

x3widget *x3vbox(x3widget *parent, int homogeneous, int spacing)
{
    GtkWidget *vbox = gtk_vbox_new(homogeneous, spacing);

    gtk_widget_show(vbox);
    return x3widget_new(parent, NULL, vbox);
}

x3widget *x3align(x3widget *parent, x3alignment alignment)
{
    int xa = alignment & 3;
    int ya = (alignment >> 2) & 3;
    float xalign = .5 * (1 + (xa >> 1) - (xa & 1));
    float yalign = .5 * (1 + (ya >> 1) - (ya & 1));
    float xscale = (xa == 3);
    float yscale = (ya == 3);
    GtkWidget *align = gtk_alignment_new(xalign, yalign, xscale, yscale);

    gtk_widget_show(align);
    return x3widget_new(parent, NULL, align);
}

x3widget *x3pad(x3widget *parent, int t, int b, int l, int r)
{
    GtkWidget *align = gtk_alignment_new(0, 0, 1, 1);

    gtk_alignment_set_padding(GTK_ALIGNMENT(align), t, b, l, r);
    gtk_widget_show(align);
    return x3widget_new(parent, NULL, align);
}

x3widget *x3button(x3widget *parent, char *cmd, char *label)
{
    GtkWidget *button = gtk_button_new_with_label(label);
    x3widget_cmdable *result = (x3widget_cmdable *)malloc(sizeof(x3widget));

    x3widget_init(&result->base, parent, cmd, button);
    result->cmd = g_strdup(cmd);
    g_signal_connect(G_OBJECT(button), "clicked",
                G_CALLBACK(x3cmdable_clicked), result);

    gtk_widget_show(button);
    return &result->base;
}

x3widget *x3edittext(x3widget *parent, char *cmd)
{
    GtkWidget *entry = gtk_entry_new();

    gtk_widget_show(entry);
    return x3widget_new(parent, cmd, entry);
}

typedef struct {
    x3widget base;
    x3viewclient *vc;
} x3widget_view;

static gboolean x3view_expose(GtkWidget *widget, GdkEventExpose *event,
                      gpointer data)
{
    x3widget_view *w = (x3widget_view *)data;

    gdk_draw_line(widget->window,
            widget->style->black_gc,
            event->area.x, event->area.y,
            event->area.x + event->area.width,
            event->area.y + event->area.height);
    return FALSE;
}

x3widget *x3view(x3widget *parent, x3viewflags flags, x3viewclient *vc)
{
    GtkWidget *event_box;
    GtkWidget *drawing_area;
    x3widget_view *result = (x3widget_view *)malloc(sizeof(x3widget_view));

    event_box = gtk_event_box_new();
    gtk_widget_show(event_box);

    drawing_area = gtk_drawing_area_new();
    gtk_container_add(GTK_CONTAINER(event_box), drawing_area);
    g_signal_connect(G_OBJECT(drawing_area), "expose_event",
                G_CALLBACK(x3view_expose), result);
    gtk_widget_show(drawing_area);

    x3widget_init(&result->base, parent, NULL, event_box);
    result->vc = vc;
    return &result->base;
```

```
}

void x3setactive(x3widget *w, int active)
{
    gtk_widget_set_sensitive(w->widget, active != 0);
}

void x3viewclient_init(x3viewclient *vc)
{
    vc->destroy = NULL;
    vc->mouse = NULL;
    vc->key = NULL;
    vc->rgb = NULL;
}
```