1. **static gboolean expose\_event(GtkWidget \*widget, GdkEventExpose \*event)**

Gtkwidget- Base class for all widgets. This function used to emit the event signals on a widget.

1. **GdkGC \*black\_gc = widget->style->black\_gc;**

GdkGC- argument, many drawing operation use it. It contains number of drawing attributes such as foreground/background color , line width

**3. gdk\_draw\_rectangle ()**

void gdk\_draw\_rectangle ([GdkDrawable](http://developer.gimp.org/api/2.0/gdk/gdk-Drawing-Primitives.html#GdkDrawable) \*drawable,

[GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC) \*gc,

[gboolean](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gboolean) filled,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) x,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) y,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) width,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) height);

Draws a rectangular outline or filled rectangle, using the foreground color and other attributes of the [GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC).

A rectangle drawn filled is 1 pixel smaller in both dimensions than a rectangle outlined. Calling gdk\_draw\_rectangle (window, gc, TRUE, 0, 0, 20, 20) results in a filled rectangle 20 pixels wide and 20 pixels high. Calling gdk\_draw\_rectangle (window, gc, FALSE, 0, 0, 20, 20) results in an outlined rectangle with corners at (0, 0), (0, 20), (20, 20), and (20, 0), which makes it 21 pixels wide and 21 pixels high.

**Note**

|  |  |
| --- | --- |
| *drawable* : | a [GdkDrawable](http://developer.gimp.org/api/2.0/gdk/gdk-Drawing-Primitives.html#GdkDrawable) (a [GdkWindow](http://developer.gimp.org/api/2.0/gdk/gdk-Windows.html#GdkWindow) or a [GdkPixmap](http://developer.gimp.org/api/2.0/gdk/gdk-Bitmaps-and-Pixmaps.html#GdkPixmap)). |
| *gc* : | a [GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC). |
| *filled* : | [TRUE](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Standard-Macros.html#TRUE:CAPS) if the rectangle should be filled. |
| *x* : | the x coordinate of the left edge of the rectangle. |
| *y* : | the y coordinate of the top edge of the rectangle. |
| *width* : | the width of the rectangle. |
| *height* : | the height of the rectangle. |

1. **gdk\_draw\_arc ()**

void gdk\_draw\_arc ([GdkDrawable](http://developer.gimp.org/api/2.0/gdk/gdk-Drawing-Primitives.html#GdkDrawable) \*drawable,

[GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC) \*gc,

[gboolean](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gboolean) filled,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) x,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) y,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) width,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) height,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) angle1,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) angle2);

Draws an arc or a filled 'pie slice'. The arc is defined by the bounding rectangle of the entire ellipse, and the start and end angles of the part of the ellipse to be drawn.

|  |  |
| --- | --- |
| *drawable* : | a [GdkDrawable](http://developer.gimp.org/api/2.0/gdk/gdk-Drawing-Primitives.html#GdkDrawable) (a [GdkWindow](http://developer.gimp.org/api/2.0/gdk/gdk-Windows.html#GdkWindow) or a [GdkPixmap](http://developer.gimp.org/api/2.0/gdk/gdk-Bitmaps-and-Pixmaps.html#GdkPixmap)). |
| *gc* : | a [GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC). |
| *filled* : | [TRUE](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Standard-Macros.html#TRUE:CAPS) if the arc should be filled, producing a 'pie slice'. |
| *x* : | the x coordinate of the left edge of the bounding rectangle. |
| *y* : | the y coordinate of the top edge of the bounding rectangle. |
| *width* : | the width of the bounding rectangle. |
| *height* : | the height of the bounding rectangle. |
| *angle1* : | the start angle of the arc, relative to the 3 o'clock position, counter-clockwise, in 1/64ths of a degree. |
| *angle2* : | the end angle of the arc, relative to *angle1*, in 1/64ths of a degree. |

1. **gdk\_draw\_line ()**

void gdk\_draw\_line ([GdkDrawable](http://developer.gimp.org/api/2.0/gdk/gdk-Drawing-Primitives.html#GdkDrawable) \*drawable,

[GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC) \*gc,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) x1\_,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) y1\_,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) x2\_,

[gint](http://developer.gimp.org/usr/share/gtk-doc/html/glib/glib-Basic-Types.html#gint) y2\_);

Draws a line, using the foreground color and other attributes of the [GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC).

|  |  |
| --- | --- |
| *drawable* : | a [GdkDrawable](http://developer.gimp.org/api/2.0/gdk/gdk-Drawing-Primitives.html#GdkDrawable) (a [GdkWindow](http://developer.gimp.org/api/2.0/gdk/gdk-Windows.html#GdkWindow) or a [GdkPixmap](http://developer.gimp.org/api/2.0/gdk/gdk-Bitmaps-and-Pixmaps.html#GdkPixmap)). |
| *gc* : | a [GdkGC](http://developer.gimp.org/api/2.0/gdk/gdk-Graphics-Contexts.html#GdkGC). |
| *x1\_* : | the x coordinate of the start point. |
| *y1\_* : | the y coordinate of the start point. |
| *x2\_* : | the x coordinate of the end point. |
| *y2\_* : | the y coordinate of the end point. |

**#include <gtk/gtk.h>**  
You need to include gtk.h.

**GtkWidget \*window;**  
All GTK objects, called "widgets" from now on, need to be defined as a pointer to a GtkWidget structure. The pointer will be allocated some memory by the use of a gtk\_WIDGET\_new() function.

**gtk\_init(&argc, &argv);**  
The first GTK function call that should be made when using GTK+ is gtk\_init(). This function takes as parameters &argc and &argv. gtk\_init() will strip off the debug switches that it wants, and leave the rest for the program. If you don't call gtk\_init() first, and you do anything using GTK+, then you get about 6 pages of failed assertions, So obviously it's doing a lot of other stuff too. Read the code for details.

**window = gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);**  
The window pointer is allocated some space by the gtk\_window\_new() function. Note that the window was defined as a GtkWidget, not a GtkWindow or something like that. The reason for this is demonstrated below. GTK\_WINDOW\_TOPLEVEL is just a constant that means you are generating a window that isn't a temporary one. The other two possible settings are GTK\_WINDOW\_DIALOG, and GTK\_WINDOW\_POPUP.

**gtk\_window\_set\_title(GTK\_WINDOW(window), "test window");**  
The window title is set by gtk\_window\_set\_title(). It requires two parameters, one is the title to set it to, naturally, but the other is the pointer to a window. Now all you have at the moment is a pointer to a GtkWidget, so It must be forced to the correct type through the use of a macro, like GTK\_WINDOW() . If the pointer is not what it should be, then an assertion will fail. (This is good for debugging. Either that or the makers of GTK+ are real bastards.)

**gtk\_widget\_show(window);**  
The window is then displayed by calling gtk\_widget\_show(). It's not necessary to cast it to a different type. If the parent of the widget you are showing isn't shown itelf yet, then you don't see anything just yet. When the parent finally is shown, the child widgets already shown will appear as well.

**gtk\_main();**  
Once you have things how you want them, gtk\_main() is called, which takes over control of the program. Once gtk\_main() returns, your program can quit gracefully. gtk\_main only returns when gtk\_main\_quit() is called. It isn't in this example, so you will notice how closing the window doesn't give you the prompt back. You need to do the two fingured salute, CTRL-C, to get it back.