

[Home](#) [Contents](#)

Basic drawing in Cairo

In this part of the Cairo graphics tutorial, we will draw some basic primitives. We will draw simple lines, use fill and stroke operations, we will talk about dashes, line caps and line joins.

Ofertas: até -70%

As melhores ofertas na sua cidade. Até 70% de desconto. Confira!
www.GROUPON.com.br/Ofertas

Ads by Google

Lines

Lines are very basic vector objects. To draw a line, we use two function calls. The starting point is specified with the `cairo_move_to()` call. The ending point of a line is specified with the `cairo_line_to()` call.

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

double coordx[100];
double coordy[100];

int count = 0;

static gboolean
on_expose_event(GtkWidget *widget,
                 GdkEventExpose *event,
                 gpointer data)
{
    cairo_t *cr;

    cr = gdk_cairo_create(widget->window);

    cairo_set_source_rgb(cr, 0, 0, 0);
    cairo_set_line_width (cr, 0.5);

    int i, j;
    for ( i = 0; i <= count - 1; i++ ) {
        for ( j = 0; j <= count -1; j++ ) {
            cairo_move_to(cr, coordx[i], coordy[i]);
            cairo_line_to(cr, coordx[j], coordy[j]);
        }
    }

    count = 0;
    cairo_stroke(cr);
    cairo_destroy(cr);
}
```

```

    return FALSE;
}

gboolean clicked(GtkWidget *widget, GdkEventButton *event,
                gpointer user_data)
{
    if (event->button == 1) {
        coordx[count] = event->x;
        coordy[count++] = event->y;
    }

    if (event->button == 3) {
        gtk_widget_queue_draw(widget);
    }

    return TRUE;
}

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

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    gtk_widget_add_events (window, GDK_BUTTON_PRESS_MASK);

    g_signal_connect(window, "expose-event",
                     G_CALLBACK(on_expose_event), NULL);
    g_signal_connect(window, "destroy",
                     G_CALLBACK(gtk_main_quit), NULL);
    g_signal_connect(window, "button-press-event",
                     G_CALLBACK(clicked), NULL);

    gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
    gtk_window_set_title(GTK_WINDOW(window), "lines");
    gtk_window_set_default_size(GTK_WINDOW(window), 400, 300);
    gtk_widget_set_app_paintable(window, TRUE);

    gtk_widget_show_all(window);

    gtk_main();

    return 0;
}

```

In our example, we click randomly on a window with a left mouse button. Each click is stored in an array. When we right click on the window, all points are connected with every point in the array. This way, we can create some interesting objects. Right clicking on the drawn object clears the window, and we can click another object.

```

cairo_set_source_rgb(cr, 0, 0, 0);
cairo_set_line_width (cr, 0.5);

```

The lines will be drawn in black ink and will be 0.5 points wide.

```
int i, j;
for ( i = 0; i <= count - 1; i++ ) {
    for ( j = 0; j <= count -1; j++ ) {
        cairo_move_to(cr, coordx[i], coordy[i]);
        cairo_line_to(cr, coordx[j], coordy[j]);
    }
}
```

We connect every point from the array to every other point.

```
cairo_stroke(cr);
```

The `cairo_stroke()` call draws the lines.

```
g_signal_connect(window, "button-press-event",
    G_CALLBACK(clicked), NULL);
```

We connect the **button-press-event** to the clicked callback.

```
if (event->button == 1) {
    coordx[count] = event->x;
    coordy[count++] = event->y;
}
```

Inside the clicked callback, we determine, if we there was a left or right click. If we click with a left mouse button, we store the x, y coordinates into the arrays.

```
if (event->button == 3) {
    gtk_widget_queue_draw(widget);
}
```

By right clicking, we redraw the window.

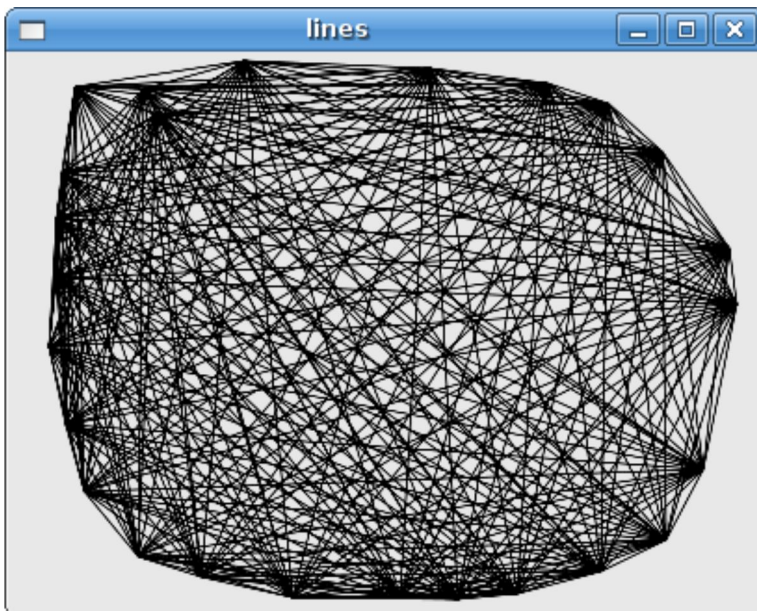


Figure: Lines

Fill and stroke

The stroke operation draws the outlines of shapes and the fill operation fills the insides of shapes.

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

static gboolean
on_expose_event (GtkWidget *widget,
                  GdkEventExpose *event,
                  gpointer data)
{
    cairo_t *cr;

    cr = gdk_cairo_create (widget->window);

    int width, height;
    gtk_window_get_size(GTK_WINDOW(widget), &width, &height);
    cairo_set_line_width(cr, 9);

    cairo_set_source_rgb(cr, 0.69, 0.19, 0);
    cairo_arc(cr, width/2, height/2,
              (width < height ? width : height) / 2 - 10, 0, 2 * M_PI);
    cairo_stroke_preserve(cr);

    cairo_set_source_rgb(cr, 0.3, 0.4, 0.6);
    cairo_fill(cr);

    cairo_destroy(cr);

    return FALSE;
}
```

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

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    g_signal_connect(G_OBJECT(window), "expose-event",
        G_CALLBACK(on_expose_event), NULL);
    g_signal_connect(G_OBJECT(window), "destroy",
        G_CALLBACK(gtk_main_quit), NULL);

    gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
    gtk_window_set_default_size(GTK_WINDOW(window), 200, 150);

    gtk_widget_set_app_paintable(window, TRUE);
    gtk_widget_show_all(window);

    gtk_main();

    return 0;
}
```

In our example, we will draw a circle and will it with a solid color.

```
#include <math.h>
```

This header file is needed for the `M_PI` constant.

```
int width, height;
gtk_window_get_size(GTK_WINDOW(widget), &width, &height);
```

Here we get the width and height of the window. We will need these values, when we draw the circle. The circle will be resized, when we resize the window.

```
cairo_set_source_rgb(cr, 0.69, 0.19, 0);
cairo_arc(cr, width/2, height/2,
    (width < height ? width : height) / 2 - 10, 0, 2 * M_PI);
cairo_stroke_preserve(cr);
```

Here we will draw the outline of a circle.

```
cairo_set_source_rgb(cr, 0.3, 0.4, 0.6);
cairo_fill(cr);
```

Here we fill the circle with blue color.

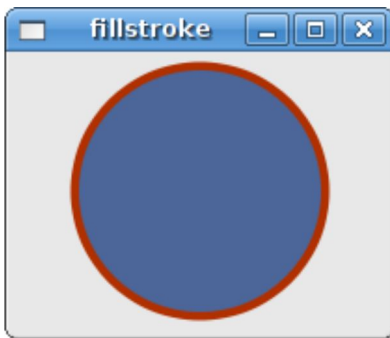


Figure: Fill and stroke

Dash

Each line can be drawn with a different pen dash. It defines the style of the line. The dash is used by the `cairo_stroke()` function call. The dash pattern is specified by the `cairo_set_dash()` function. The pattern is set by the dash array, which is an array of positive values. They set the on and off parts of the dash pattern. We also specify the length of the array and the offset value. If the length is 0, the dashing is disabled. If it is 1, a symmetric pattern is assumed. The offset value specifies, where the dashing begins. Or in other words, it is an empty space at the beginning.

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

static gboolean
on_expose_event(GtkWidget *widget,
                 GdkEventExpose *event,
                 gpointer data)
{
    cairo_t *cr;

    cr = gdk_cairo_create (widget->window);

    cairo_set_source_rgba(cr, 0, 0, 0, 1);

    static const double dashed1[] = {4.0, 1.0};
    static int len1 = sizeof(dashed1) / sizeof(dashed1[0]);

    static const double dashed2[] = {4.0, 1.0, 4.0};
    static int len2 = sizeof(dashed2) / sizeof(dashed2[0]);

    static const double dashed3[] = {1.0};

    cairo_set_line_width(cr, 1.5);

    cairo_set_dash(cr, dashed1, len1, 0);

    cairo_move_to(cr, 40, 30);
    cairo_line_to(cr, 200, 30);
    cairo_stroke(cr);
```

```
    cairo_set_dash(cr, dashed2, len2, 1);

    cairo_move_to(cr, 40, 50);
    cairo_line_to(cr, 200, 50);
    cairo_stroke(cr);

    cairo_set_dash(cr, dashed3, 1, 0);

    cairo_move_to(cr, 40, 70);
    cairo_line_to(cr, 200, 70);
    cairo_stroke(cr);

    cairo_destroy(cr);

    return FALSE;
}

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

    GtkWidget *window;
    GtkWidget *darea;

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    darea = gtk_drawing_area_new();
    gtk_container_add(GTK_CONTAINER(window), darea);

    g_signal_connect(darea, "expose-event",
                     G_CALLBACK (on_expose_event), NULL);
    g_signal_connect(window, "destroy",
                     G_CALLBACK (gtk_main_quit), NULL);

    gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
    gtk_window_set_default_size(GTK_WINDOW(window), 250, 100);

    gtk_widget_show_all(window);

    gtk_main();

    return 0;
}
```

In this example, we will draw three lines with different dash patterns.

```
static const double dashed1[] = {4.0, 1.0};
```

Here we specify our first dash pattern. It will draw a pattern of 4 pixels and one empty point.

```
static int len1 = sizeof(dashed1) / sizeof(dashed1[0]);
```

We get the size of the array.

```
cairo_set_dash(cr, dashed1, len1, 0);
```

We set the dash.

```
darea = gtk_drawing_area_new();  
gtk_container_add(GTK_CONTAINER(window), darea);
```

In this example, we do not draw directly on the window but on the drawing area.



Figure: Dashes

Line caps

The line caps are endpoints of lines.

- CAIRO_LINE_CAP_SQUARE
- CAIRO_LINE_CAP_ROUND
- CAIRO_LINE_CAP_BUTT

There are three different line cap styles in Cairo.



Figure: Square, round and butt caps

A line with a `CAIRO_LINE_CAP_SQUARE` cap will have a different size, than a line with a `CAIRO_LINE_CAP_BUTT` cap. If a line is width `px` wide, the line with a `CAIRO_LINE_CAP_SQUARE` cap will be exactly width `px` greater in size. `width/2 px` at the beginning and `width/2 px` at the end.

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



```
static gboolean
on_expose_event(GtkWidget *widget,
    GdkEventExpose *event,
    gpointer data)
{
    cairo_t *cr;

    cr = gdk_cairo_create (widget->window);

    cairo_set_source_rgba(cr, 0, 0, 0, 1);
    cairo_set_line_width(cr, 10);

    cairo_set_line_cap(cr, CAIRO_LINE_CAP_BUTT);
    cairo_move_to(cr, 30, 50);
    cairo_line_to(cr, 150, 50);
    cairo_stroke(cr);

    cairo_set_line_cap(cr, CAIRO_LINE_CAP_ROUND);
    cairo_move_to(cr, 30, 90);
    cairo_line_to(cr, 150, 90);
    cairo_stroke(cr);

    cairo_set_line_cap(cr, CAIRO_LINE_CAP_SQUARE);
    cairo_move_to(cr, 30, 130);
    cairo_line_to(cr, 150, 130);
    cairo_stroke(cr);

    cairo_set_line_width(cr, 1.5);

    cairo_move_to(cr, 30, 40);
    cairo_line_to(cr, 30, 140);
    cairo_stroke(cr);

    cairo_move_to(cr, 150, 40);
    cairo_line_to(cr, 150, 140);
    cairo_stroke(cr);

    cairo_move_to(cr, 155, 40);
    cairo_line_to(cr, 155, 140);
    cairo_stroke(cr);

    cairo_destroy(cr);

    return FALSE;
}

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

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);
```

```
darea = gtk_drawing_area_new ();
gtk_container_add(GTK_CONTAINER (window), darea);

g_signal_connect(darea, "expose-event",
    G_CALLBACK(on_expose_event), NULL);
g_signal_connect(window, "destroy",
    G_CALLBACK(gtk_main_quit), NULL);

gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
gtk_window_set_default_size(GTK_WINDOW(window), 200, 200);

gtk_widget_show_all(window);

gtk_main();

return 0;
}
```

The example draws three lines with three different caps. It will also graphically demonstrate the differences in size of the lines.

```
cairo_set_line_width(cr, 10);
```

Our lines will be 10 px wide.

```
cairo_set_line_cap(cr, CAIRO_LINE_CAP_ROUND);
cairo_move_to(cr, 30, 90);
cairo_line_to(cr, 150, 90);
cairo_stroke(cr);
```

Here we draw a horizontal line with a CAIRO_LINE_CAP_ROUND cap.

```
cairo_set_line_width(cr, 1.5);

cairo_move_to(cr, 30, 40);
cairo_line_to(cr, 30, 140);
cairo_stroke(cr);
```

This is one of the three vertical lines used to demonstrate the differences in size.

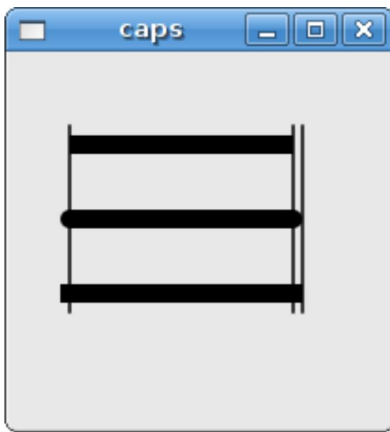


Figure: Line caps

Line joins

The lines can be joined using three different join styles.

- CAIRO_LINE_JOIN_MITER
- CAIRO_LINE_JOIN_BEVEL
- CAIRO_LINE_JOIN_ROUND

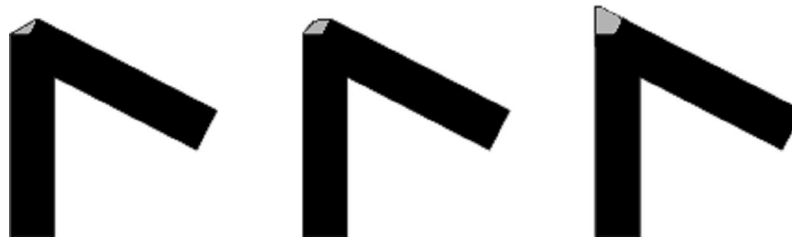


Figure: Bevel, Round, Miter line joins

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

static gboolean
on_expose_event(GtkWidget *widget,
    GdkEventExpose *event,
    gpointer data)
{
    cairo_t *cr;

    cr = gdk_cairo_create (widget->window);

    cairo_set_source_rgb(cr, 0.1, 0, 0);

    cairo_rectangle(cr, 30, 30, 100, 100);
    cairo_set_line_width(cr, 14);
    cairo_set_line_join(cr, CAIRO_LINE_JOIN_MITER);
    cairo_stroke(cr);
}
```

```
cairo_rectangle(cr, 160, 30, 100, 100);
cairo_set_line_width(cr, 14);
cairo_set_line_join(cr, CAIRO_LINE_JOIN_BEVEL);
cairo_stroke(cr);

cairo_rectangle(cr, 100, 160, 100, 100);
cairo_set_line_width(cr, 14);
cairo_set_line_join(cr, CAIRO_LINE_JOIN_ROUND);
cairo_stroke(cr);

cairo_destroy(cr);

return FALSE;
}

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

    gtk_init(&argc, &argv);

    window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

    darea = gtk_drawing_area_new();
    gtk_container_add(GTK_CONTAINER(window), darea);

    g_signal_connect(darea, "expose-event",
        G_CALLBACK(on_expose_event), NULL);
    g_signal_connect(window, "destroy",
        G_CALLBACK(gtk_main_quit), NULL);

    gtk_window_set_position(GTK_WINDOW(window), GTK_WIN_POS_CENTER);
    gtk_window_set_default_size(GTK_WINDOW(window), 300, 280);

    gtk_widget_show_all(window);

    gtk_main();


    return 0;
}
```

In this example, we draw three thick rectangles with various line joins.

```
cairo_rectangle(cr, 30, 30, 100, 100);
cairo_set_line_width(cr, 14);
cairo_set_line_join(cr, CAIRO_LINE_JOIN_MITER);
cairo_stroke(cr);
```

In this code example, we draw a rectangle with `CAIRO_LINE_JOIN_MITER` join style. The lines are 14 px wide.

joins



[Draw Floor Plans](#)
[Fast](#)
Floor Plan
Drawing Software
See Examples.
Free Download!
www.SmartDraw.com

ns

we did some basic drawing.

[tour privado a Israel](#)
paquete de lujo a Israel-Egipto Excursion de lujo en Israel-Egipto
www.ahalanolympus.co.il/

Ads by Google

[Home](#) † [Contents](#) † [Top of Page](#)