Guile-GNOME: PangoCairo

version 2.16.2, updated 9 December 2011

Owen Taylor
Behdad Esfahbod
many others

Short Contents

1 Overview

`(gnome pangocairo)` wraps the Cairo integration layer of the Pango text rendering library for guile. It is a part of Guile-GNOME.

See the documentation for `(gnome gobject)` for more information on Guile-GNOME.

2 Cairo Rendering

Rendering with the Cairo backend

2.1 Overview

The Cairo library (`http://cairographics.org`) is a vector graphics library with a powerful rendering model. It has such features as anti-aliased primitives, alpha-compositing, and gradients. Multiple backends for Cairo are available, to allow rendering to images, to PDF files, and to the screen on X and on other windowing systems. The functions in this section allow using Pango to render to Cairo surfaces.

Using Pango with Cairo is straightforward. A `<pango-context>` created with `pango-cairo-font-map-create-context` can be used on any Cairo context (cairo_t), but needs to be updated to match the current transformation matrix and target surface of the Cairo context using `pango-cairo-update-context`. The convenience functions `pango-cairo-create-layout` and `pango-cairo-update-layout` handle the common case where the program doesn't need to manipulate the properties of the `<pango-context>`.

When you get the metrics of a layout or of a piece of a layout using functions such as `pango-layout-get-extents`, the reported metrics are in user-space coordinates. If a piece of text is 10 units long, and you call cairo_scale (cr, 2.0), it still is more-or-less 10 units long. However, the results will be affected by hinting (that is, the process of adjusting the text to look good on the pixel grid), so you shouldn't assume they are completely independent of the current transformation matrix. Note that the basic metrics functions in Pango report results in integer Pango units. To get to the floating point units used in Cairo divide by '`PANGO_SCALE`'.

```
#include <math.h>
#include <pango/pangocairo.h>

static void
draw_text (cairo_t *cr)
{
#define RADIUS 150
#define N_WORDS 10
#define FONT "Sans Bold 27"

  PangoLayout *layout;
  PangoFontDescription *desc;
  int i;

  /* Center coordinates on the middle of the region we are drawing
   */
  cairo_translate (cr, RADIUS, RADIUS);

  /* Create a PangoLayout, set the font and text */
  layout = pango_cairo_create_layout (cr);
```

```
  pango_layout_set_text (layout, "Text", -1);
  desc = pango_font_description_from_string (FONT);
  pango_layout_set_font_description (layout, desc);
  pango_font_description_free (desc);

  /* Draw the layout N_WORDS times in a circle */
  for (i = 0; i < N_WORDS; i++)
    {
      int width, height;
      double angle = (360. * i) / N_WORDS;
      double red;

      cairo_save (cr);

      /* Gradient from red at angle == 60 to blue at angle == 240 */
      red   = (1 + cos ((angle - 60) * G_PI / 180.)) / 2;
      cairo_set_source_rgb (cr, red, 0, 1.0 - red);

      cairo_rotate (cr, angle * G_PI / 180.);

      /* Inform Pango to re-layout the text with the new transformation */
      pango_cairo_update_layout (cr, layout);

      pango_layout_get_size (layout, &width, &height);
      cairo_move_to (cr, - ((double)width / PANGO_SCALE) / 2, - RADIUS);
      pango_cairo_show_layout (cr, layout);

      cairo_restore (cr);
    }

  /* free the layout object */
  g_object_unref (layout);
}

int main (int argc, char **argv)
{
  cairo_t *cr;
  char *filename;
  cairo_status_t status;
  cairo_surface_t *surface;

  if (argc != 2)
    {
      g_printerr ("Usage: cairosimple OUTPUT_FILENAME\n");
      return 1;
    }
```

```
  filename = argv[1];

  surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32,
2 * RADIUS, 2 * RADIUS);
  cr = cairo_create (surface);


  cairo_set_source_rgb (cr, 1.0, 1.0, 1.0);
  cairo_paint (cr);
  draw_text (cr);
  cairo_destroy (cr);

  status = cairo_surface_write_to_png (surface, filename);
  cairo_surface_destroy (surface);

  if (status != CAIRO_STATUS_SUCCESS)
    {
      g_printerr ("Could not save png to '%s'\n", filename);
      return 1;
    }

  return 0;
}
```

2.2 Usage

`<pango-cairo-font-map>` [Class]

Derives from `<ginterface>`.

This class defines no direct slots.

`pango-cairo-font-map-new` ⇒ (*ret* `<pango-font-map>`) [Function]

Creates a new `<pango-cairo-font-map>` object; a fontmap is used to cache information about available fonts, and holds certain global parameters such as the resolution. In most cases, you can use `pango-cairo-font-map-get-default` instead.

Note that the type of the returned object will depend on the particular font backend Cairo was compiled to use; You generally should only use the `<pango-font-map>` and `<pango-cairo-font-map>` interfaces on the returned object.

ret the newly allocated `<pango-font-map>`, which should be freed with `g-object-unref`.

Since 1.10

`pango-cairo-font-map-get-default` ⇒ (*ret* `<pango-font-map>`) [Function]

Gets a default font map to use with Cairo.

ret the default Cairo fontmap for `<pango>`. This object is owned by Pango and must not be freed.

Since 1.10

`pango-cairo-font-map-set-resolution` [Function]
(*self* `<pango-cairo-font-map>`) (*dpi* `double`)

`set-resolution` [Method]

Sets the resolution for the fontmap. This is a scale factor between points specified in a `<pango-font-description>` and Cairo units. The default value is 96, meaning that a 10 point font will be 13 units high. (10 * 96. / 72. = 13.3).

fontmap a `<pango-cairo-font-map>`

dpi the resolution in `"dots per inch"`. (Physical inches aren't actually involved; the terminology is conventional.)

Since 1.10

`pango-cairo-font-map-get-resolution` [Function]
(*self* `<pango-cairo-font-map>`) ⇒ (*ret* `double`)

`get-resolution` [Method]

Gets the resolution for the fontmap. See `pango-cairo-font-map-set-resolution`

fontmap a `<pango-cairo-font-map>`

ret the resolution in `"dots per inch"`

Since 1.10

`pango-cairo-font-map-create-context` [Function]
(*self* `<pango-cairo-font-map>`) ⇒ (*ret* `<pango-context>`)

`create-context` [Method]

Create a `<pango-context>` for the given fontmap.

fontmap a `<pango-cairo-font-map>`

ret the newly created context; free with `g-object-unref`.

Since 1.10

`pango-cairo-context-set-resolution` (*context* `<pango-context>`) [Function]
(*dpi* `double`)

Sets the resolution for the context. This is a scale factor between points specified in a `<pango-font-description>` and Cairo units. The default value is 96, meaning that a 10 point font will be 13 units high. (10 * 96. / 72. = 13.3).

context a `<pango-context>`, from `pango-cairo-font-map-create-context`

dpi the resolution in `"dots per inch"`. (Physical inches aren't actually involved; the terminology is conventional.) A 0 or negative value means to use the resolution from the font map.

Since 1.10

`pango-cairo-context-get-resolution` (*context* `<pango-context>`) [Function]
⇒ (*ret* `double`)

Gets the resolution for the context. See `pango-cairo-context-set-resolution`

context a `<pango-context>`, from `pango-cairo-font-map-create-context` *ret* the resolution in "dots per inch". A negative value will be returned if no resolution has previously been set.

Since 1.10

`pango-cairo-update-context` (*cr* `cairo-t`) (*context* `<pango-context>`) [Function]

Updates a `<pango-context>` previously created for use with Cairo to match the current transformation and target surface of a Cairo context. If any layouts have been created for the context, it's necessary to call `pango-layout-context-changed` on those layouts.

cr a Cairo context

context a `<pango-context>`, from `pango-cairo-font-map-create-context`

Since 1.10

`pango-cairo-create-layout` (*cr* `cairo-t`) ⇒ (*ret* `<pango-layout>`) [Function]

Creates a layout object set up to match the current transformation and target surface of the Cairo context. This layout can then be used for text measurement with functions like `pango-layout-get-size` or drawing with functions like `pango-cairo-show-layout`. If you change the transformation or target surface for *cr*, you need to call `pango-cairo-update-layout`

This function is the most convenient way to use Cairo with Pango, however it is slightly inefficient since it creates a separate `<pango-context>` object for each layout. This might matter in an application that was laying out large amounts of text.

cr a Cairo context

ret the newly created `<pango-layout>`. Free with `g-object-unref`.

Since 1.10

`pango-cairo-update-layout` (*cr* `cairo-t`) (*layout* `<pango-layout>`) [Function]

Updates the private `<pango-context>` of a `<pango-layout>` created with `pango-cairo-create-layout` to match the current transformation and target surface of a Cairo context.

cr a Cairo context

layout a `<pango-layout>`, from `pango-cairo-create-layout`

Since 1.10

`pango-cairo-show-glyph-string` (*cr* `cairo-t`) (*font* `<pango-font>`) (*glyphs* `<pango-glyph-string>`) [Function]

Draws the glyphs in *glyphs* in the specified cairo context. The origin of the glyphs (the left edge of the baseline) will be drawn at the current point of the cairo context.

cr a Cairo context

font a `<pango-font>`

glyphs a `<pango-glyph-string>`

Since 1.10

`pango-cairo-show-layout-line` (*cr* `cairo-t`) (*line* `<pango-layout-line>`) [Function]

Draws a `<pango-layout-line>` in the specified cairo context. The origin of the glyphs (the left edge of the line) will be drawn at the current point of the cairo context.

cr a Cairo context

line a `<pango-layout-line>`

Since 1.10

`pango-cairo-show-layout` (*cr* `cairo-t`) (*layout* `<pango-layout>`) [Function]

Draws a `<pango-layout-line>` in the specified cairo context. The top-left corner of the `<pango-layout>` will be drawn at the current point of the cairo context.

cr a Cairo context

layout a Pango layout

Since 1.10

`pango-cairo-show-error-underline` (*cr* `cairo-t`) (*x* `double`) (*y* `double`) (*width* `double`) (*height* `double`) [Function]

Draw a squiggly line in the specified cairo context that approximately covers the given rectangle in the style of an underline used to indicate a spelling error. (The width of the underline is rounded to an integer number of up/down segments and the resulting rectangle is centered in the original rectangle)

cr a Cairo context

x The X coordinate of one corner of the rectangle

y The Y coordinate of one corner of the rectangle

width Non-negative width of the rectangle

height Non-negative height of the rectangle

Since 1.14

`pango-cairo-glyph-string-path` (*cr* `cairo-t`) (*font* `<pango-font>`) (*glyphs* `<pango-glyph-string>`) [Function]

Adds the glyphs in *glyphs* to the current path in the specified cairo context. The origin of the glyphs (the left edge of the baseline) will be at the current point of the cairo context.

cr a Cairo context

font a `<pango-font>`

glyphs a `<pango-glyph-string>`

Since 1.10

`pango-cairo-layout-line-path` (*cr* `cairo-t`) (*line* `<pango-layout-line>`) [Function]

Adds the text in `<pango-layout-line>` to the current path in the specified cairo context. The origin of the glyphs (the left edge of the line) will be at the current point of the cairo context.

cr a Cairo context

line a <pango-layout-line>

Since 1.10

pango-cairo-layout-path (*cr* cairo-t) (*layout* <pango-layout>) [Function]

Adds the text in a <pango-layout> to the current path in the specified cairo context. The top-left corner of the <pango-layout> will be at the current point of the cairo context.

cr a Cairo context

layout a Pango layout

Since 1.10

pango-cairo-error-underline-path (*cr* cairo-t) (*x* double) (*y* double) (*width* double) (*height* double) [Function]

Add a squiggly line to the current path in the specified cairo context that approximately covers the given rectangle in the style of an underline used to indicate a spelling error. (The width of the underline is rounded to an integer number of up/down segments and the resulting rectangle is centered in the original rectangle)

cr a Cairo context

x The X coordinate of one corner of the rectangle

y The Y coordinate of one corner of the rectangle

width Non-negative width of the rectangle

height Non-negative height of the rectangle

Since 1.14

3 Undocumented

The following symbols, if any, have not been properly documented.

3.1 (gnome gw pangocairo)

`pango-cairo-context-get-font-options` [Function]

`pango-cairo-context-set-font-options` [Function]

Type Index

Function Index

C

G

P

S