renderpdf

Rendering a PDF or PS file with cairo

PDF files can be rendered to a cairo context using poppler. PS or EPS files can also be rendered to a cairo context by first converting to PDF using Ghostscript.

When using a vector backend, the vectors and text in the PDF file are preserved in the output as vectors. There is no unnecessary rasterization.

Compile the examples with:

```
gcc -o pdftops pdftops.c `pkg-config --cflags --libs cairo poppler-glib`
gcc -o pdftoimage pdftoimage.c `pkg-config --cflags --libs cairo poppler-glib`
```

pdftops.c - Render a PDF file to a cairo PostScript surface.

```
#include <poppler.h>
#include <cairo.h>
#include <cairo-ps.h>

int main(int argc, char *argv[])
{
    PopplerDocument *document;
    PopplerPage *page;
    double width, height;
    GError *error;
    const char *filename;
    gchar *absolute, *uri;
    int num_pages, i;
    cairo_surface_t *surface;
    cairo_t *cr;
    cairo_status_t status;

    if (argc != 2) {
        printf ("Usage: pdf2cairo input_file.pdf\n");
        return 0;
    }

    filename = argv[1];
    g_type_init ();
```

```
    error = NULL;

    if (g_path_is_absolute(filename)) {
        absolute = g_strdup (filename);
    } else {
        gchar *dir = g_get_current_dir ();
        absolute = g_build_filename (dir, filename, (gchar *) 0);
        free (dir);
    }

    uri = g_filename_to_uri (absolute, NULL, &error);
    free (absolute);
    if (uri == NULL) {
        printf("poppler fail: %s\n", error->message);
        return 1;
    }

    document = poppler_document_new_from_file (uri, NULL, &error);
    if (document == NULL) {
        printf("poppler fail: %s\n", error->message);
        return 1;
    }

    num_pages = poppler_document_get_n_pages (document);

    /* Page size does not matter here as the size is changed before
     * each page */
    surface = cairo_ps_surface_create ("output.ps", 595, 842);
    cr = cairo_create (surface);
    for (i = 0; i < num_pages; i++) {
        page = poppler_document_get_page (document, i);
        if (page == NULL) {
            printf("poppler fail: page not found\n");
            return 1;
        }
        poppler_page_get_size (page, &width, &height);
        cairo_ps_surface_set_size (surface, width, height);
        cairo_save (cr);
        poppler_page_render_for_printing (page, cr);
        cairo_restore (cr);
        cairo_surface_show_page (surface);
        g_object_unref (page);
    }
    status = cairo_status(cr);
    if (status)
        printf("%s\n", cairo_status_to_string (status));
    cairo_destroy (cr);
    cairo_surface_finish (surface);
    status = cairo_surface_status(surface);
    if (status)
        printf("%s\n", cairo_status_to_string (status));
```

```
    cairo_surface_destroy (surface);

    g_object_unref (document);

    return 0;
}
```

pdftoimage.c - Render a one page of a PDF file to a cairo image surface.

```
#include <stdio.h>
#include <stdlib.h>
#include <poppler.h>
#include <cairo.h>

#define IMAGE_DPI 150

int main(int argc, char *argv[])
{
    PopplerDocument *document;
    PopplerPage *page;
    double width, height;
    GError *error;
    const char *pdf_file;
    const char *png_file;
    gchar *absolute, *uri;
    int page_num, num_pages;
    cairo_surface_t *surface;
    cairo_t *cr;
    cairo_status_t status;

    if (argc != 4) {
        printf ("Usage: pdftoimage input_file.pdf output_file.png page\n");
        return 0;
    }

    pdf_file = argv[1];
    png_file = argv[2];
    page_num = atoi(argv[3]);
    g_type_init ();
    error = NULL;

    if (g_path_is_absolute(pdf_file)) {
        absolute = g_strdup (pdf_file);
    } else {
        gchar *dir = g_get_current_dir ();
        absolute = g_build_filename (dir, pdf_file, (gchar *) 0);
        free (dir);
    }

    uri = g_filename_to_uri (absolute, NULL, &error);
    free (absolute);
```

```
    if (uri == NULL) {
        printf("%s\n", error->message);
        return 1;
    }

    document = poppler_document_new_from_file (uri, NULL, &error);
    if (document == NULL) {
        printf("%s\n", error->message);
        return 1;
    }

    num_pages = poppler_document_get_n_pages (document);
    if (page_num < 1 || page_num > num_pages) {
        printf("page must be between 1 and %d\n", num_pages);
        return 1;
    }

    page = poppler_document_get_page (document, page_num - 1);
    if (page == NULL) {
        printf("poppler fail: page not found\n");
        return 1;
    }

    poppler_page_get_size (page, &width, &height);

    /* For correct rendering of PDF, the PDF is first rendered to a
     * transparent image (all alpha = 0). */
    surface = cairo_image_surface_create (CAIRO_FORMAT_ARGB32,
                                          IMAGE_DPI*width/72.0,
                                          IMAGE_DPI*height/72.0);
    cr = cairo_create (surface);
    cairo_scale (cr, IMAGE_DPI/72.0, IMAGE_DPI/72.0);
    cairo_save (cr);
    poppler_page_render (page, cr);
    cairo_restore (cr);
    g_object_unref (page);

    /* Then the image is painted on top of a white "page". Instead of
     * creating a second image, painting it white, then painting the
     * PDF image over it we can use the CAIRO_OPERATOR_DEST_OVER
     * operator to achieve the same effect with the one image. */
    cairo_set_operator (cr, CAIRO_OPERATOR_DEST_OVER);
    cairo_set_source_rgb (cr, 1, 1, 1);
    cairo_paint (cr);

    status = cairo_status(cr);
    if (status)
        printf("%s\n", cairo_status_to_string (status));

    cairo_destroy (cr);
    status = cairo_surface_write_to_png (surface, png_file);
```

```
    if (status)
        printf("%s\n", cairo_status_to_string (status));

    cairo_surface_destroy (surface);

    g_object_unref (document);

    return 0;
}
```

Last edited Mon Apr 25 19:01:59 2016