Using cairo library to create PDF files

Adam Graliński

C++ FFFE, May 2021

Cairo

- open source 2D graphics library
  - GNU Lesser General Public License (LGPL) 2.1 or Mozilla Public License (MPL) 1.1
- support for multiple output devices
  - *stable*: X Window System, Win32, image buffers, PostScript files, PDF files, SVGs
  - *experimental*: OpenGL, BeOS (Haiku), DirectFB
- provides subroutines for drawing primitives, stroking and filling, transforming, rendering text, etc.
- mature and stable project
  - initial release before 2013
  - latest release (1.17.4) 29.11.2020, second-latest (1.17.2) 01.02.2019,
  - Written in C

https://www.cairographics.org

Simple usage

1. create a drawing surface
2. create cairo context
3. stroke some curves, render some text, fill some polygon, etc.
4. render the page
5. destroy the allocated resources

```
#include <cairo.h>
#include <cairo-pdf.h>
#include <cstdlib>
#include <iostream>

int main()
{
  cairo_surface_t* csurface =
      cairo_pdf_surface_create("file.pdf", 800, 600); // 1
  if (cairo_surface_status(csurface) != CAIRO_STATUS_SUCCESS) {
    std::cerr << "Could␣not␣generate␣cairo␣surface\n";
    std::exit(1);
  } else {
    std::cout << "Successfully␣created␣cairo␣surface\n";
  }
  cairo_t* cr = cairo_create(csurface); // 2

  // 3

  cairo_show_page(cr); // 4
  cairo_surface_destroy(csurface); // 5
  cairo_destroy(cr);               // 5
}
```

Listing 1: creating an empty PDF page (*listings/01.cpp*)

Sidenote: typographic points

- Cairo measures surfaces and primitives in *typographic points*.
- in typography, 1 inch = 72 tp
- use converting functions if you prefer metric system

```
constexpr inline double mm_to_tp(double mm) { return mm / 25.4 * 72; }
```

- A4 page is 210×297 millimeters wide.

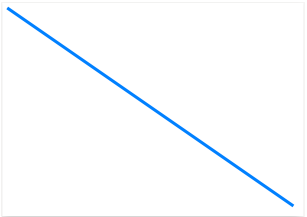
Drawing a line

Let's draw *a line*:

1. set pen width
   `cairo_set_line_width(cr, width);`
   https://www.cairographics.org/manual/cairo-cairo-t.html#cairo-set-line-width
2. set pen color
   `cairo_set_source_rgb(cr, red, green, blue);`
   https://www.cairographics.org/manual/cairo-cairo-t.html#cairo-set-source-rgb
   note: the values for red, green and blue should be in range of [0.0 ... 1.0]
3. use `cairo_move_to(cr, x, y)` and `cairo_line_to(cr, x, y)` to draw a line
   https://www.cairographics.org/manual/cairo-Paths.html#cairo-move-to
   https://www.cairographics.org/manual/cairo-Paths.html#cairo-line-to
4. finally call `cairo_stroke(cr)` to stroke the path — this draws the line
   https://www.cairographics.org/manual/cairo-cairo-t.html#cairo-stroke

Drawing a line

```
#include <cairo.h>
#include <cairo-pdf.h>
#include <cstdlib>
#include <iostream>

constexpr inline double mm_to_tp(double mm) { return mm / 25.4 * 72; }
const double A4_WIDTH = mm_to_tp(297);
const double A4_HEIGHT = mm_to_tp(210);

int main()
{
  cairo_surface_t* csurface =
      cairo_pdf_surface_create("file.pdf", A4_WIDTH, A4_HEIGHT); // 1
  if (cairo_surface_status(csurface) != CAIRO_STATUS_SUCCESS) {
    std::cerr << "Could not generate cairo surface\n";
    std::exit(1);
  } else {
    std::cout << "Successfully created cairo surface\n";
  }
  cairo_t* cr = cairo_create(csurface); // 2

  // 3
  cairo_set_source_rgb(cr, 0.0, 0.5, 1.0);
  cairo_set_line_width(cr, mm_to_tp(3));
  cairo_move_to(cr, mm_to_tp(5), mm_to_tp(5));
  cairo_line_to(cr, A4_WIDTH - mm_to_tp(10), A4_HEIGHT - mm_to_tp(10));
  cairo_stroke(cr);

  cairo_show_page(cr); // 4
  cairo_surface_destroy(csurface); // 5
  cairo_destroy(cr);               // 5
}
```



expected output

Listing 2: drawing a blue line (*listings/02.cpp*)

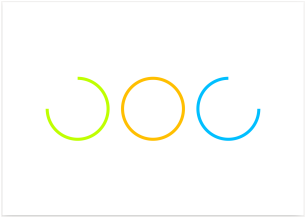
Drawing arcs and circles

Let's draw *some circles*:

1. set pen width
   `cairo_set_line_width(cr, width);`
   https://www.cairographics.org/manual/cairo-cairo-t.html#cairo-set-line-width
2. set pen color
   `cairo_set_source_rgb(cr, red, green, blue);`
   https://www.cairographics.org/manual/cairo-cairo-t.html#cairo-set-source-rgb
   note: the values for red, green and blue should be in range of [0.0 ... 1.0]
3. draw an arc
   `cairo_arc(cr, center_x, center_y, radius, angle_start, angle_end)`
   https://www.cairographics.org/manual/cairo-Paths.html#cairo-arc
   note: angles are measured in radians
4. finally call `cairo_stroke(cr)` to stroke the path — this draws the arcs
   https://www.cairographics.org/manual/cairo-cairo-t.html#cairo-stroke

Drawing arcs and circles

```
#include <cairo.h>
#include <cairo-pdf.h>
#include <cmath>
#include <iostream>

constexpr inline double mm_to_tp(double mm) { return mm / 25.4 * 72; }
const double A4_WIDTH = mm_to_tp(297);
const double A4_HEIGHT = mm_to_tp(210);

int main()
{
  cairo_surface_t* csurface =
      cairo_pdf_surface_create("file.pdf", A4_WIDTH, A4_HEIGHT); // 1
  if (cairo_surface_status(csurface) != CAIRO_STATUS_SUCCESS) {
    std::cerr << "Could not generate cairo surface\n";
    std::exit(1);
  } else {
    std::cout << "Successfully created cairo surface\n";
  }
  cairo_t* cr = cairo_create(csurface); // 2

  // 3
  cairo_set_line_width(cr, mm_to_tp(3));

  cairo_set_source_rgb(cr, 0.75, 1.0, 0.0);
  cairo_arc(cr, 0.25 * A4_WIDTH, 0.5 * A4_HEIGHT,
      mm_to_tp(30), 1.5 * M_PI, 3.0 * M_PI);
  cairo_stroke(cr);

  cairo_set_source_rgb(cr, 1.0, 0.75, 0.0);
  cairo_arc(cr, 0.5 * A4_WIDTH, 0.5 * A4_HEIGHT,
      mm_to_tp(30), 0.0, 2.0 * M_PI);
  cairo_stroke(cr);

  cairo_set_source_rgb(cr, 0.0, 0.75, 1.0);
  cairo_arc(cr, 0.75 * A4_WIDTH, 0.5 * A4_HEIGHT,
      mm_to_tp(30), 0.0 * M_PI, 1.5 * M_PI);
  cairo_stroke(cr);

  cairo_show_page(cr); // 4
  cairo_surface_destroy(csurface); // 5
  cairo_destroy(cr);               // 5
}
```



expected output

Listing 3: drawing arcs (*listings/03.cpp*)

Rendering text

Let's draw *a caption*:

1. set font face
   ```
   cairo_select_font_face(cr, font, slant, weight);
   ```
   https://www.cairographics.org/manual/cairo-text.html#cairo-select-font-face
2. set font size
   ```
   cairo_set_font_size(cr, size);
   ```
   https://www.cairographics.org/manual/cairo-text.html#cairo-set-font-size
3. move the pen to the chosen position
   ```
   cairo_move_to(cr, x, y)
   ```
4. render the text
   ```
   cairo_show_text(cr, text);
   ```
   https://www.cairographics.org/manual/cairo-text.html#cairo-show-text

Rendering text

```
#include <cairo.h>
#include <cairo-pdf.h>
#include <cmath>
#include <iostream>

constexpr inline double mm_to_tp(double mm) { return mm / 25.4 * 72; }
const double A4_WIDTH = mm_to_tp(297);
const double A4_HEIGHT = mm_to_tp(210);

int main()
{
  cairo_surface_t* csurface =
      cairo_pdf_surface_create("file.pdf", A4_WIDTH, A4_HEIGHT); // 1
  if (cairo_surface_status(csurface) != CAIRO_STATUS_SUCCESS) {
    std::cerr << "Could␣not␣generate␣cairo␣surface\n";
    std::exit(1);
  } else {
    std::cout << "Successfully␣created␣cairo␣surface\n";
  }
  cairo_t* cr = cairo_create(csurface); // 2

  // 3
  cairo_select_font_face(cr, "serif",
      CAIRO_FONT_SLANT_NORMAL, CAIRO_FONT_WEIGHT_NORMAL);
  cairo_set_font_size(cr, 42);
  cairo_move_to(cr, 0.25 * A4_WIDTH, 0.5 * A4_HEIGHT);
  cairo_show_text(cr, "Hello,␣World␣of␣PDFs!");

  cairo_show_page(cr); // 4
  cairo_surface_destroy(csurface); // 5
  cairo_destroy(cr);               // 5
}
```

Listing 4: rendering a line of text (*listings/04.cpp*)

expected output

Real-life examples

(demo)

Key takeaways

- it's very easy to just get started with Cairo
- SVG images are supported too
- cairo text API is called a "toy API" for good reasons
  - combine Cairo with Pango library if you need precise control over text rendering
- use cairomm for full C++/STL support
- compile all listings with: `g++ -std=c++20 ‘pkg-config --cflags --libs cairo‘ 0X.cpp`

Thank you!