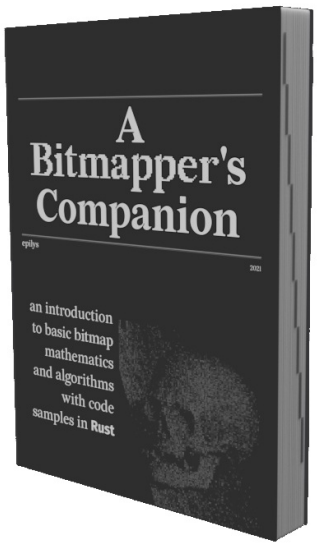
“A Bitmap Companion”

open source print book about raster graphics

This project started as notes on simple raster graphics algorithms (like drawing a line segment) that I was compiling while I was working on another project*: a remake of the classic `x11` kitkat clock† that uses bitmap sprites of a cat and draws a clock and a swinging tail manually. I decided to do it in **Rust** and use the `minifb` crate to draw directly at a framebuffer window. Then I realised I don't know how to do graphics, and perhaps I wasn't paying much attention at Linear Algebra class in university. Seeing as my note sources were scattered across books and the internet, I started compiling them in a LaTeX document.



A simulacrum (i.e. render) of what a printed version would look like. Thank you *Blender v3.0*.

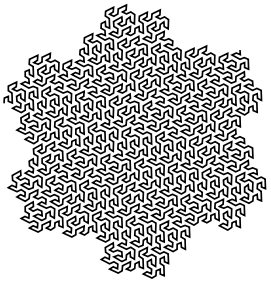
The book has one chapter per graphics task, with math explanations when needed and a code example. Many of the code examples have a corresponding **Rust** binary in the crate included with the book, and unlike the book these can be interactive. I had great fun when drawing quadratic Bézier curves because I could manipulate their control points on the framebuffer with the mouse. Other binaries like the line intersection one are also interactive (e.g. you can move around the lines and compute the new intersection).

The `attachfile2` package is used to attach **Rust** source files in the pdf.

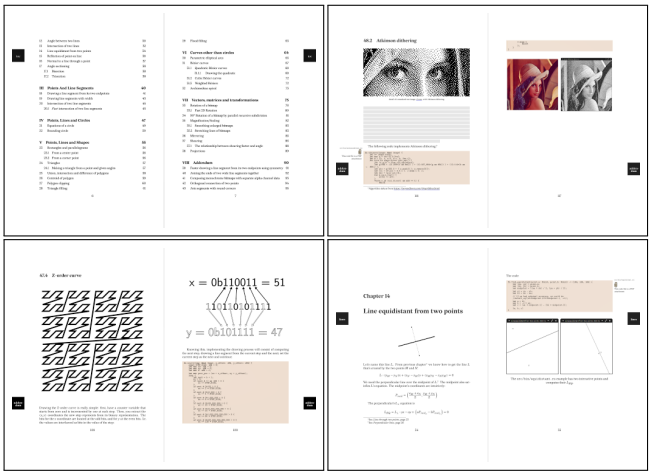
Most of the book's figures are made by hand in Inkscape. For complex drawings like the space filling curves, I edited the code examples to output SVGs by emitting a `path` element everytime a line segment is drawn in the framebuffer.

```
    let mut output = vec![];
    output.push(format!(
        r#"<svg width="{}" height="{}"
↪   xmlns="http://www.w3.org/2000/svg">"#,
        img.width
        img.height
    ));
    ...
      // For every bitmap line segment (x1, y1), (x2,
↪   y2):
      output.push(format!(
              r#"  <path d="M {} {} L {} {}"
↪   stroke="black" fill="none"/>"#,
              x1, y1, x2, y2
      ));
    ...
    output.push("</svg>".to_string());
    println!("{}", output.join("\n"));
```

At the program's exit, I can print all the paths to `stdout` and take a not necessarily good SVG. For the Gosper curve, for example, I had to manually select all the non-endpoint points of the curve and join them in order to produce a single discrete path:



The order 4 flowsnake curve SVG.



Page samples

As you can see in the page samples, some of the book's paragraphs are blank placeholders, courtesy of the `skeldoc` package. I used them to fill each chapter so that I could plan out the book's structure before I have everything written down and ready.

The book's LaTeX code and the **Rust** code are all licensed under **GPLv3**, and the text content under **cc-nc-sa-3.0**. The latest source code and pdf build will always be available at `https://github.com/epilys/bitmappers-companion`. I hope when it's ready an actual print release becomes possible. – *epilys*

*`https://crates.io/crates/kitkat`

†`https://github.com/plan9foundation/plan9/blob/main/sys/src/games/catclock.c`