Using Asymptote like METAPOST

Jim Hefferon

Abstract

Asymptote is a vector graphics language for technical drawing that fits very well with TEX, LATEX, and friends. It deserves to be more widely known.

One appealing thing is that it is in part based on algorithms from METAFONT and METAPOST but it extends those to three dimensions. I'll discuss a couple of workflow issues that a beginner to Asymptote who is coming from METAPOST might find useful, in particular using a single source file to output many related graphics.

1 Introduction

I wrote a book years ago using METAPOST and found that it had many advantages. I cannot draw, at all, and it was a comfort to be able to tell the computer to, say, make this line to be exactly two thirds the length of that other line.

But the best feature is that METAPOST fits a person who thinks mathematically. For example there is a simple way to find where two lines intersect.

However, having worked with METAPOST a lot, I was aware of some warts. For me the two biggest are lack of any real 3D abilities, and that programming in the language can be ... quirky. So when I saw the new Asymptote system I was eager to try it. It has been very good.

2 Overview

Asymptote is a powerful descriptive vector graphics language. It provides a natural coordinate-based framework for technical drawing. LATEX typesets the text and equations.[1]

- It is inspired by METAPOST, including generalizing METAPOST's path construction algorithms to three dimensions.[2] It inherits METAPOST's impedance match with a mathematical mindset. But it is a more standard programming language, including declared types, familiar syntax, and IEEE floating point numbers.
- It outputs high-quality PostScript, OpenGL, PDF, SVG, WebGL, and V3D, as well as 3D vector WebGL graphics for HTML files and 3D vector PRC graphics for PDF files.
- It uses deferred drawing to solve size constraint issues between fixed-sized objects, such as labels and arrowheads, and objects that should scale with figure size.
- It runs on Unix, macOS, and Windows, and is under continuing development.

You can even try it in your browser without installing it, using the Asymptote Web Application.[3]

3 Compared to Ti*k*Z

Many readers will be familiar with Ti*k*Z so I will briefly compare with that system.

I have not used Ti*k*Z much, and have not used it lately at all. But I lined up the two when I was starting my latest project, some years ago. They have many of the same strengths. I found that Asymptote had some technical advantages, including the native 3D graphics, and I also found Ti*k*Z harder to program in.

Another difference, relevant here, is that the basic paradigm in Ti*k*Z is that your figures are in your document, generated when the document is generated. In Asymptote the paradigm is that they are generated outside the document. (Yes, you can generate stand-alone in Ti*k*Z and yes, you can include Asymptote code in a document.)

I had a bad experience in the past when I was using PSTricks and the LATEX world switched to pdfLATEX. That left me with a preference for a dependence chain that is shorter and broader over one that is taller and thinner. I saw this again recently when there was an issue with Ghostscript and I was able to put the external PDF figures into my repository until the issue was resolved. So one factor in my comparison tipping me to Asymptote was a preference for generating figures independently from the documents.

4 Working in METAPOST

The basic structure of METAPOST input is that one file holds many figures. This will output a graphic into a file numbered 1 and another graphic into a file numbered 2.

```
beginfig (1)  % MetaPost
  % first figure drawing commands
endfig;
beginfig (2)
  % second figure drawing commands
endfig;
```

Putting a number of related sources in the same file is convenient. For instance, if these are drawings for a calculus lecture then you might at the top of the file declare `VECTOR_THICKNESS=0.8pt`, and use that in lots of the figures.

[1] asymptote.sourceforge.io/asymptote.pdf
[2] asymptote.sourceforge.io/gallery/3Dgraphs/
[3] http://asymptote.ualberta.ca/

Jim Hefferon
doi.org/10.47397/tb/44-2/tb137hefferon-asymptote

I will describe two adjustments that may help a person coming from METAPOST and that took me some time to dope out.

5 Adjustment one: multiple figures per file

Here is the skeleton to put multiple figures in one Asymptote file.

This outputs a file `test000.pdf` containing the graphic with a diagonal red line.

```
string OUTPUT_FN = "test%03d";
// =============
picture pic; int picnum = 0;
unitsize(pic, 1cm);  // dist from x=0 to x=1
draw(pic, (0,0)--(1,1), red); // make a line
shipout(format(OUTPUT_FN,picnum),
         pic, format="pdf");

// =============
picture pic; int picnum = 1;
  ...
shipout(...);
```

Some things to notice in that code:

- The `OUTPUT_FN` gives all file names the same structure. So if you have lots of graphics (my current book has more than 2000) then it is easier to work with them. Of course, `%03d` gives the picture number as a three decimal place integer. I find two decimal places is too tight.
- A second picture begins with the declarations `picture pic` and `int picnum = 1` and ends with an identical `shipout(...)`, as shown.
- I write `picnum = 0` and `picnum = 1`, etc., instead of `picnum = picnum+1`. When you go back a week later into a file with eighty pictures, looking to fix a bug in the fifty-third, you want it this way.
- Because of the multiple outputs from a single input, lots of commands need a picture argument. The code has it in the `draw(...)` command. If you leave out the `pic` then the line gets drawn somewhere (that is, there is no error) but not in the output file where you are looking for it. The `pic` is also in the `unitsize(...)` command.

A comment: METAPOST is set up so the figure code blocks are isolated, in that if, say, you set `x=1` inside one `beginfig ... endfig` then that does not affect an `x` inside another. There is no such isolation here. But I sometimes reuse variables in a number of related figures so I'm happy with this.

6 Adjustment two: style files

This is related to the prior adjustment in that one way that having multiple outputs from a single input is helpful is to enforce uniformity. You can have parameters such as line thickness or font size, and specify them in just the one file.

But the same applies across multiple files. It is useful to have every Asymptote source file get desired global parameters from a single file.

You bring in a file with the `import fn` command. For instance, Asymptote has a standard file called `settings.asy` that you usually want to bring in.

```
import settings;
settings.outformat="pdf";
```

If that file is in the list of directories searched by the Asymptote system then you are good. The list is what you might guess: first the current directory, then one or more directories given by the environment variable `ASYMPTOTE_DIR`, etc.

Here is part of the style file for my current book.

```
import fontsize;
defaultpen(fontsize(9.24994pt));
import texcolors;
pen darkgrey_color=rgb("595241");
pen lightgrey_color=rgb("E0D4BE");
pen white_color=rgb("FFFFFF");
pen lightblue_color=rgb("ACCFCC");
pen red_color=rgb("8A0917");
// Use these names, not prior ones
pen highlightcolor=red_color;
pen backgroundcolor=lightblue_color;
pen boldcolor=darkgrey_color;
pen lightcolor=lightgrey_color;
pen verylightcolor=white_color;
```

(I like the color names such as `highlightcolor` to make graphics because I sometimes adjust the color scheme, for instance when I get a proof copy from the publisher.)

In my current project I also have a LaTeX style file containing the font information and document-specific macros that is used by every Asymptote file, as well as by the LaTeX driver file `book.tex`, ensuring that the figures match the text.

7 Ending

Asymptote does a great job drawing technical graphics. Adding some METAPOST-like workflow makes it even more fun.

◇ Jim Hefferon
Mathematics and Statistics
University of Vermont
jim.hefferon (at) gmail dot com
https://hefferon.net