Tools for Getting Graphics Into LaTeX

Part 2

Dan Drake

Korea Advanced Institute of Science and Technology

14 August 2008 / Sage Days 9

Outline

Outline

Good resources

Links below are all clickable, so all you *really* need to know is that these slides will be at `wiki.sagemath.org/DanDrake/Days9Talks`.

- books: I like *A Guide to LaTeX* and *The LaTeX Companion*
- *The LaTeX Graphics Companion*. Be sure you have the 2nd edition.
- *The PracTeX Journal*, `www.tug.org/pracjourn`
- the web, LaTeX geeks

Outline

Strategy *i*: use something else to create the graphics

- xfig (`xfig.org`)
- Inkscape (`inkscape.org`)
- PiScript (`www.google.ca/search?q=piscript`)

Also...

- Dia (`www.gnome.org/projects/dia`)
- Gnuplot (`www.gnuplot.info`)
- GIMP (`www.gimp.org`)
- tons of others (`google.com` and `www.maa.org/editorial/mathgames/mathgames_08_01_05.html`)

Oh, one more:

Sage!

sagemath.org

Using xfig optimally

If you add text

- select the “special” text flag
- change the text font to a LaTeX font

Then use “combined PS/PDF/LaTeX (3 parts)” to export your graphics. Choose a filename, and do `\input{filename}` in your document.

Advantage: text rendered at compile time, so your document can be compiled using different fonts than you have. No extra stuff needed by anyone else who has all your files.

Using xfig optimally

If you add text

- select the “special” text flag
- change the text font to a LaTeX font

Then use “combined PS/PDF/LaTeX (3 parts)” to export your graphics. Choose a filename, and do `\input{filename}` in your document.

Advantage: text rendered at compile time, so your document can be compiled using different fonts than you have. No extra stuff needed by anyone else who has all your files.

Using `psfrag`

The `psfrag` package accomplishes much the same thing as `xfig`'s combined export:

- put a text tag ("Rn") into your EPS file
- do `\psfrag{Rn}{\mathcal{R}^n}`
- do usual `\includegraphics` stuff

Any "Rn" in your EPS files are magically replaced by $\mathcal{R}^n$.
Unfortunately, you can't use `pdflatex`; you need to use `dvips`.

See also: `psfragx`, `overpic` (other LaTeX packages)

Render text as paths

Inkscape (via the `textext` extension) and PiScript take your LaTeX and render it into a big path. This is nice, but you do need to re-render if you change anything about the font in your document and want text in your images to change too.

But. . . anyone who can use EPS graphics can see the text exactly as you intended.

Render text as paths

Inkscape (via the `textext` extension) and PiScript take your LaTeX and render it into a big path. This is nice, but you do need to re-render if you change anything about the font in your document and want text in your images to change too.

But. . . anyone who can use EPS graphics can see the text exactly as you intended.

Outline

Strategy 1: use LaTeX to create the graphics

Two big players: PSTricks (`tug.org/PSTricks`) and PGF/Ti*k*Z (`www.google.com/search?q=tikz`). PSTricks is older and better known, but I think Ti*k*Z has many advantages.

Two other well-known packages worthy of consideration: XY-pic, which is designed for commutative diagrams, and MetaPost, which is a separate language/file format but is part of the TeX ecosystem.

Strategy 1: use LaTeX to create the graphics

Two big players: PSTricks (`tug.org/PSTricks`) and PGF/Ti*k*Z (`www.google.com/search?q=tikz`). PSTricks is older and better known, but I think Ti*k*Z has many advantages.

Two other well-known packages worthy of consideration: XY-pic, which is designed for commutative diagrams, and MetaPost, which is a separate language/file format but is part of the TeX ecosystem.

PSTricks

PSTricks is in the same spirit as PiScript: sortof an interface to Postscript from TEX.

The "PS" means Postscript, so compiling to a PDF requires `dvips` and `ps2pdf`. But you can do more or less anything that Postscript can.

PSTricks

PSTricks is in the same spirit as PiScript: sortof an interface to Postscript from TEX.

The “PS” means Postscript, so compiling to a PDF requires `dvips` and `ps2pdf`. But you can do more or less anything that Postscript can.

PGF and Ti*k*Z

PGF stands for “Portable Graphics Format” and is the underlying engine for the higher-level Ti*k*Z, which stands for “Ti*k*Z ist kein Zeichenprogramm”.

Ti*k*Z works with plain TEX, LATEX, ConTEXt, compiling to DVI or PDF. It comes with a well-written 560-page manual.

Ti*k*Z syntax, though still complicated, is often more consistent than PSTricks, as Ti*k*Z was designed knowing the shortcomings of PSTricks. Pretty much the only thing Ti*k*Z can’t do that PSTricks can is text that follows a path.

PGF and Ti*k*Z

PGF stands for “Portable Graphics Format” and is the underlying engine for the higher-level Ti*k*Z, which stands for “Ti*k*Z ist kein Zeichenprogramm”.

Ti*k*Z works with plain TEX, LATEX, ConTEXt, compiling to DVI or PDF. It comes with a well-written 560-page manual.

Ti*k*Z syntax, though still complicated, is often more consistent than PSTricks, as Ti*k*Z was designed knowing the shortcomings of PSTricks. Pretty much the only thing Ti*k*Z can’t do that PSTricks can is text that follows a path.

PGF and Ti*k*Z

PGF stands for “Portable Graphics Format” and is the underlying engine for the higher-level Ti*k*Z, which stands for “Ti*k*Z ist kein Zeichenprogramm”.

Ti*k*Z works with plain TEX, LATEX, ConTEXt, compiling to DVI or PDF. It comes with a well-written 560-page manual.

Ti*k*Z syntax, though still complicated, is often more consistent than PSTricks, as Ti*k*Z was designed knowing the shortcomings of PSTricks. Pretty much the only thing Ti*k*Z can’t do that PSTricks can is text that follows a path.

Good features of PSTricks and Ti*k*Z

Both work as *building blocks* (higher level or specialized packages are built on them) and as *output formats*—several graphical programs output PSTricks or Ti*k*Z code, and there's a Python script that converts xfig files to Ti*k*Z.

My recommendations

- Choose PSTricks or Ti*k*Z and use that for as much of your inline graphics as possible. I prefer Ti*k*Z.
- Use Sage, Inkscape, xfig, and psfrag for external graphics. Use SageTEX (`tug.ctan.org/pkg/sagetex`) to help integrate Sage into your document.

Support for Ti*k*Z is not as widespread as PSTricks (arXiv!) but see "externalizing graphics" in the PGF/Ti*k*Z manual. *See next slide for correction!*

My recommendations

- Choose PSTricks or Ti*k*Z and use that for as much of your inline graphics as possible. I prefer Ti*k*Z.
- Use Sage, Inkscape, xfig, and psfrag for external graphics. Use SageTEX (`tug.ctan.org/pkg/sagetex`) to help integrate Sage into your document.

Support for Ti*k*Z is not as widespread as PSTricks (arXiv!) but see "externalizing graphics" in the PGF/Ti*k*Z manual. *See next slide for correction!*

Correction, 20 August 2008

I discovered today that the previous slide's claim about the arXiv is *wrong*: they *do* support in submissions. See arxiv:0708.0245 for a paper that uses for figures.

The future: TEX and SVG

SVG is “Scalable Vector Graphics”. Designed for use on the web, but browser support is still sub-optimal. Very promising, though.

There’s a `dvi2svg` converter, but more work needs to be done. I think this is an important development, though, so stay tuned.

Thank you

These slides (source and PDF) will be available from the Sage Days 9 wiki page, and `wiki.sagemath.org/DanDrake/Days9Talks`

Outline

Special director's cut bonus scenes!

Here is some exclusive bonus material, available only here. I've added this after my talk, because in giving these talks I learned a lot from my audience. (It often works that way, doesn't it?)

- Michael Abshoff mentioned jfig: `http://tams-www.informatik.uni-hamburg.de/applets/jfig/`. It does work, but it looks like the nice LaTeX export facilities depend on the `fig2dev` utility. . . which I thinks works in Windows.
- The cool SVG blobs I showed: `www.themaninblue.com/experiment/Blobular/`
- An article on `dvi2svg`, ironically (though unsurprisingly) in PDF format: `tug.org/TUGboat/Articles/tb27-2/tb87frischauf.pdf` and here are some examples: `wmula.republika.pl/proj/pydvi2svg/index.html#samples`

More bonus material. . .

Bonus scenes continued

- The cool Ti*k*Z animation:
  `www.fauskes.net/pgftikzexamples/animated-distributions/`.
  Nils Bruin found out more out using LaTeX for animations:
  `wiki.sagemath.org/days9/PDFLaTeXAnimations`