The graphviz package*

Derek Rayside ⟨drayside@mit.edu⟩
with contributions from Ralf Hemmecke ⟨ralf@hemmecke.de⟩

August 16, 2006

1 Introduction

graphviz.sty is a LaTeX package for writing graphviz/dot/neato graphs inside of LaTeX documents. graphviz.sty was inspired by a feature that Daniel Jackson added to his tagger text markup tool.

graphviz is a freely available package for doing automated graph layout from AT&T Research, distributed under the Common Public License (CPL). graphviz includes the dot and neato programs, which read a textual description of a graph and produces a graphical rendering of it. Many different graphics formats, include PostScript, are supported.

There are two main web pages for the graphviz project:

- `http://www.graphviz.org`
- `http://www.research.att.com/sw/tools/graphviz/`

graphviz.sty is provided as-is, with no warranty or claim to fitness for any purpose, use at your own risk, etc. graphviz.sty is distributed under the LaTeX Project Public License.

2 Example

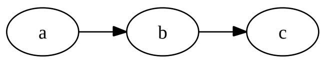
Put this in your document:

```
\digraph[scale=0.5]{abc}{rankdir=LR; a->b->c;}
```

Run these commands (only the first run needs `-shell-escape`):

```
latex -shell-escape main.tex
latex main.tex
```

And here's what you get:



*This document corresponds to graphviz v0.9, dated 2006/01/08.

3 Usage

`\digraph[⟨i⟩]{⟨n⟩}{⟨g⟩}` The `\digraph` (dot) and `\neatograph` (neato) commands take three arguments:

[⟨i⟩] parameters to the `\includegraphics` command that will include the PostScript file of the graph [this is optional]: eg, '`scale=0.5`'

{⟨n⟩} the name of the graph; a file `name.dot` is created, and a file `name.ps` is expected to be produced from dot: eg, '`MyGraph`'
{⟨n⟩} has to be a valid file name and a valid identifier name.

{⟨g⟩} the graph, specified in the dot/graphviz language:
eg, '`rankdir=LR; a->b->c;`'

4 Options

`singlefile` LaTeX has a small number of file handles (about 16 or so). So if you can't have too many digraphs in your tex file before you run out of file handles. The `singlefile` option is a work-around: it writes all of your digraphs to a single file (`master.graphviz`), and then uses gvpr to split that file into individual dot files for processing by dot.

gvpr does not seem to be packaged with the Windows version of dot.

```
\newif\ifsinglefile
\DeclareOption{singlefile}{
    \singlefiletrue
    \AtBeginDocument{ % open a new file handle
        \newwrite\masterdotfile
        \immediate\openout\masterdotfile=master.graphviz}
    \AtEndDocument{ % close the file
        \immediate\closeout\masterdotfile}}
```

`psfrag` The `psfrag` option uses the psfrag package to enable you to overlay TeX fragments over included postscript files, such as those generated via the `\digraph` command.

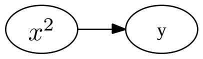
The ladot script from Brighten Godfrey uses Perl to extend the syntax of the graphviz language with TeX fragments, and psfrag to super-impose those fragments.

The `psfrag` option requires sed. psfrag seems to only work with dvips: ie, it is not compatible with pdflatex or dvipdfm. The PDF files produced by LaTeX/psfrag/ps2pdf seem to view ok with Acrobat, but not with gv. Oddly, the PS files produced this way work in gv.

Put this in your document:

```
\psfrag{x2}[cc][cc]{$x^2$}
\digraph{xy}{rankdir=LR; x2->y;}
```

And here's what you get:



```
\newif\ifpsfrag
\DeclareOption{psfrag}{ \psfragtrue }
```

Set the default options

```
\ExecuteOptions{}
\ProcessOptions\relax % LaTeX class guide says it is wise to relax
```

5 Implementation

5.1 Required Packages

This package requires graphicx to include PostScript renderings of graphs.

```
\RequirePackage{graphicx}
\ifpsfrag \RequirePackage{psfrag} \fi
```

5.2 Command Implementation

`\digraph` This is the command the user uses for dot.

It is very important that this command is not defined with 3 parameters although it will be used with 3 parameters in the form `\digraph[OPTIONS]{FILENAME}{GRAPH}`. The reason is that the catcode for `^^M` must be changed *before* TEX reads the GRAPH argument.

The order of the command (first `\inputdigraph` then `\@digraph`) may look a bit odd, but it simplifies the code. In order to include the digraph, LATEX has to be run at least two times anyway. In the first run the file dot will be generated and only the second run the digraph will be included.

```
\newcommand{\digraph}[2][scale=1]{
  \inputdigraph[#1]{#2}{dot}%       % Include the digraph.
  \@digraph{#2}%                    % Generate the .dot file.
}
```

`\neatograph` This is the command the user uses for neato.

```
\newcommand{\neatograph}[2][scale=1]{
  \inputdigraph[#1]{#2}{neato}%     % Include the digraph.
  \@digraph{#2}%                    % Generate the .dot file.
}
```

`\@digraph` Internal implementation.

The macro \@digraph prepares the actual output of the digraph to a file (which is done by \@@digraph) by a special treatment of the newline character. Before entering \@@digraph, the input newline character (^^M) is made active, and redefined to expand to ^^J. Note that \@digraph has a \begingroup that is closed in \@@digraph.

The purpose of this is to preserve line breaks in the digraph.

```
\begingroup
  \catcode`\^^M=\active%
  \gdef\@digraph{\begingroup\catcode`\^^M=\active\def^^M{^^J}\@@digraph}%
\endgroup
```

\@@digraph Internal implementation.

The parameters of the macro \@@digraph are the FILENAME and GRAPH of the initial \digraph[OPTIONS]{FILENAME}{GRAPH}. Note that if \@@digraph is entered the ^^M character is active. Thus every newline character (^^M) in the following macro is hidden through a % sign at the end of line.

```
\def\@@digraph#1#2{%
    \ifsinglefile% write the digraph to the master file
        \expandafter\def\csname -\endcsname{\string\n}%
        \immediate\write\masterdotfile{digraph #1 {#2}}%
        \write18{gvpr -o #1.dot 'BEG_G { if ($.name == "#1") {write($);} }' master.graphviz }%
    \else% open a new file handle
        \newwrite\dotfile%
        \immediate\openout\dotfile=#1.dot%
        \expandafter\def\csname -\endcsname{\string\n}%
        \immediate\write\dotfile{digraph #1 {#2}}%
        \immediate\closeout\dotfile%
    \fi%
% Here comes the closing \endgroup that closes the group opened in \@digraph.
    \endgroup}%
% Now ^^M is no longer active.
```

\inputdigraph This is usually only called by \digraph, but may be called by the user.

The purpose is to include the PostScript rendering of the graph if it exists, or to give instructions on how to generate it.

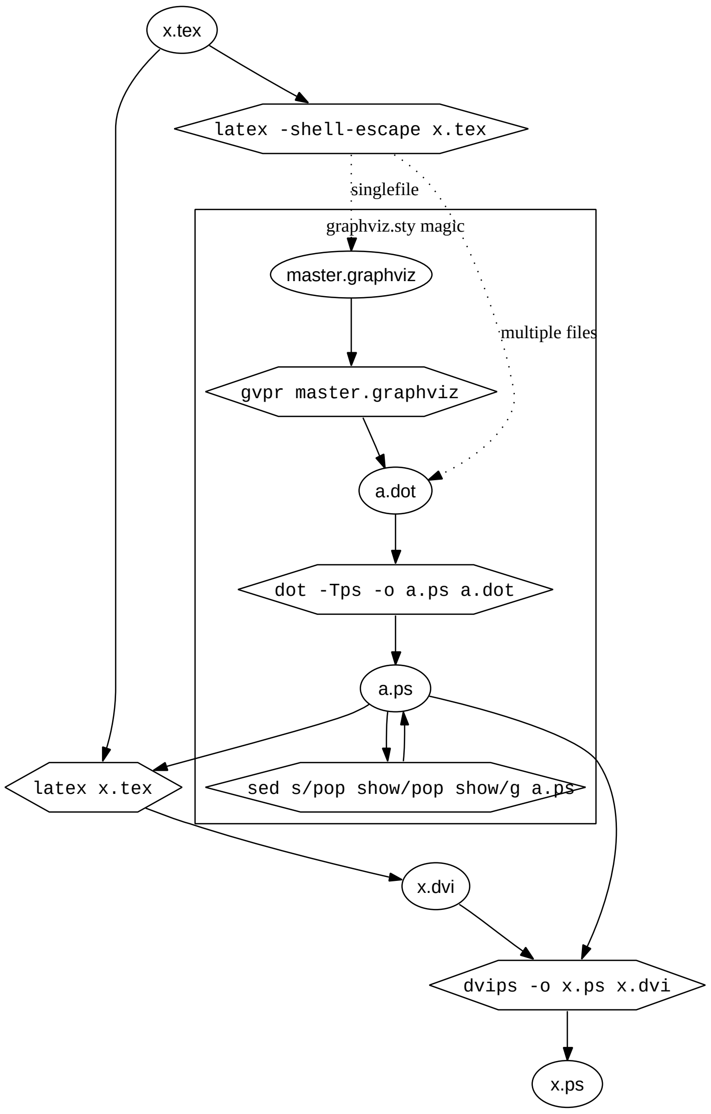
```
\newcommand{\inputdigraph}[3][scale=1]{
    % execute dot (nb: requires latex -shell-escape)
    \write18{#3 -Tps -o #2.ps #2.dot}
    \IfFileExists{#2.ps}{ % the postscript exists: include it
            \ifpsfrag
                % per the ladot 2.2 source code, psfrag has a problem with
                % graphviz 2.2, and some sed hackery is necessary to work around
                \write18{sed -ibackup -e "s/xshow/pop show/g" #2.ps}
            \fi
            \includegraphics[#1]{#2.ps}
        }
        % else: the postscript doesn't exist: tell the user how to create it
        { \fbox{ \begin{tabular}{l}
            The file \texttt{#2.ps} hasn't been created from \texttt{#2.dot}
            yet. \\ Run ‘\texttt{dot -Tps -o #2.ps #2.dot}’ to create it. \\
            Or invoke \LaTeX\ with the \texttt{-shell-escape} option
            to have this done automatically. \\
            \end{tabular}}
        }
}
```

5.3 Process

`\digraph` writes out a `dot` file, and then invokes `dot` on it.

Note: `\digraph` can only invoke `dot` if the LaTeX was invoked with the `-shell-escape` option, to enable execution of external programs. If you do not want to allow LaTeX to execute external programs, then you will have to invoke `dot` yourself. `graphviz` will also need to execute `gvpr` if the `singlefile` option has been selected, and `sed` if the `psfrag` option has been selected.

Here's a picture of the process (drawn with `dot`, naturally):


x.tex
latex -shell-escape x.tex
singlefile
graphviz.sty magic
master.graphviz
multiple files
gvpr master.graphviz
a.dot
dot -Tps -o a.ps a.dot
a.ps
latex x.tex
sed s/pop show/pop show/g a.ps
x.dvi
dvips -o x.ps x.dvi
x.ps

Change History

v0.1

General: Initial version 1

v0.2

`\digraph`: minor adjustments 3

`\inputdigraph`: minor adjustments 4

v0.4

General: converted to dtx format . 1

`\digraph`: new comments 3

v0.5

General: renamed package to dotla 1

`\digraph`: added automatic invocation of dot 3

v0.6

`\digraph`: added singlefile option . 3

`singlefile`: added singlefile option 2

v0.7

General: renamed package back to graphviz 1

`\digraph`: added backslash-hyphen line breaks by Ralf Hemmecke . 3

now using gvpr instead of gawk to break out individual digraphs from master.graphviz 3

removed redundant invocation of dot from digraph; only inputdigraph needs to invoke dot 3

`singlefile`: now using gvpr instead of gawk to break out individual digraphs from master.graphviz . 2

v0.8

`\inputdigraph`: added psfrag support 4

`psfrag`: added psfrag option 2

v0.9

`\digraph`: refactored for control-M by Ralf Hemmecke 3

`\neatograph`: added support for neato 3