Introduction to LaTeX

Week 07: Graphics and TikZ

2026-11-26

Table of Contents

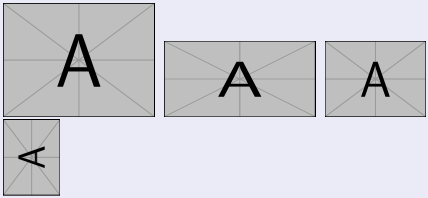
1. Graphics Files
2. TikZ Graphics
3. Best Practices
4. Errors and Obsolete Practices

Including Graphics: graphicx.sty

Input

```
\usepackage{graphicx} % in the preamble
\includegraphics[width = 2cm]{example-image-a}
\includegraphics[width = 2cm, height = 1cm]
  {example-image-a}
\includegraphics[width = 2cm, height = 1cm,
  keepaspectratio]{example-image-a}
\includegraphics[width = 1cm, angle = 90]
  {example-image-a}
```

Output

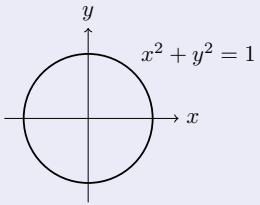


- Width and height specifications together may distort the image. `keepaspectratio` keeps the proportions so that none of width or height are exceeded.
- You may omit the file extension: pdflatex then tries `.pdf`, `.png`, `.jpg` and a few more, in this order, and takes the first it finds.
- `\graphicspath{{images/}{figures/}}` in the preamble adds folders in which to look for images, in this order.

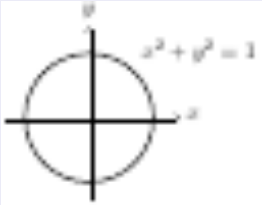
Packages in the 'graphics' Bundle

Raster and Vector Graphics

Vector graphics (PDF), enlarged



Raster graphics (PNG), enlarged



- Vector graphics store lines and letters, and stay sharp at any size.
- Raster graphics store pixels: PNG uses lossless compression, JPG employs lossy compression that suits photos.

Which format to use

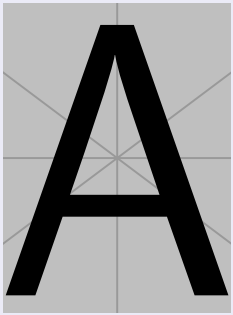
Save plots and diagrams (if you must) as PDF, so that they stay sharp in print and on screen. Use JPG for photos and PNG for screenshots. Never screenshot a plot.

Trimming an Image: `trim` and `clip`

Input

```
\includegraphics[width = 3cm,
  trim = 4cm 2cm 4cm 2cm, clip]  % l, b, r, t
  {example-image-a}
```

Output



- `trim` takes four lengths off the left, bottom, right and top edge, in this order, measured on the image at its original size. `width` and `height` applies to what is left.
- `trim` alone only shrinks the space reserved for the image, but it is still drawn in full, over the text around it. `clip` hides what lies outside that space.

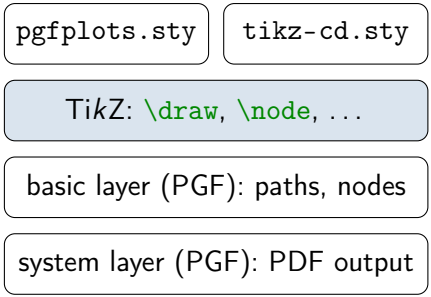
The hidden part of a clipped image is still embedded in the PDF and can be extracted. If this is undesired, the image file itselt must be cropped before LaTeX processing.

Packages in the 'graphics' Bundle

Table of Contents

1. Graphics Files
2. TikZ Graphics
3. Best Practices
4. Errors and Obsolete Practices

PGF, TikZ and the Packages Built on Them

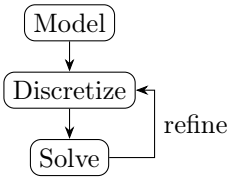


- PGF, the "portable graphics format", does the drawing underneath.
- TikZ is the language on top of it, in which pictures are written.
- `pgfplots.sty` for plots and `tikz-cd.sty` for commutative diagrams are built on TikZ.
- `\usepackage{tikz}` loads PGF as well.

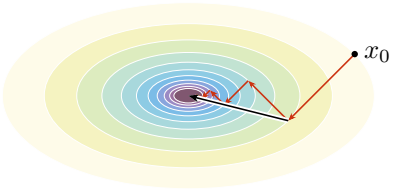
TikZ = TikZ ist kein Zeichenprogramm

Tantau, *The TikZ and PGF Packages*, 2025, p. 27

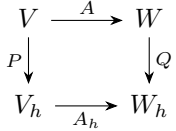
Some Examples of What TikZ Can Do



Flowcharts



Mathematical illustrations



Commutative diagrams
(`tikz-cd.sty`)

- There are numerous libraries and packages built on TikZ, e. g., to draw trees, circuits (`circuitikz.sty`), and chemical formulas (`chemfig.sty`).
- The TikZ manual is over 1300 pages long and starts with tutorials.

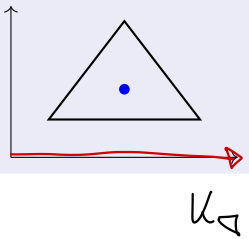
Reading TikZ Code: Paths

Input

```
\usepackage{tikz} % in the preamble

\begin{tikzpicture}
  \draw[->] (0, 0) -- (3, 0);  % hor. axis
  \draw[->] (0, 0) -- (0, 2);  % vert. axis
  \draw[thick] (0.5, 0.5) -- (2.5, 0.5)
    -- (1.5, 1.8) -- cycle;  % triangle
  \fill[blue] (1.5, 0.9) circle (2pt);
\end{tikzpicture}
```

Output



- A path starts with a command such as `\draw` or `\fill` and ends with a semicolon.
- Coordinates `(x, y)` are in centimeters unless otherwise specified.
- `--` draws a line, `cycle` closes the path.
- Options in brackets, such as `->`, `thick` or `blue`, change how a path looks.

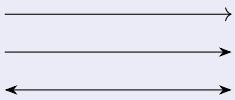
TikZ Libraries: \usetikzlibrary

Input

```
% in the preamble
\usepackage{tikz}
\usetikzlibrary{arrows.meta}

\begin{tikzpicture}
  \draw[->] (0, 1) -- (3, 1);
  \draw[-Stealth] (0, 0.5) -- (3, 0.5);
  \draw[Stealth-Stealth] (0, 0) -- (3, 0);
\end{tikzpicture}
```

Output



- A library adds features to TikZ, here the arrow tip `Stealth` and many other arrow features.
- A library is loaded in the preamble, after `tikz.sty`.

- Several libraries can go into one line, separated by commas: `\usetikzlibrary{arrows.meta, calc, positioning}`.
- Often used libraries include: `arrows.meta`, `positioning` (placing nodes relative to each other), `calc` (computing coordinates).

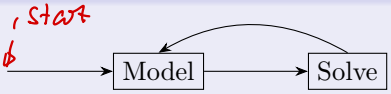
Nodes and Named Coordinates

Input

```
% in the preamble
\usepackage{tikz}
\usetikzlibrary{arrows.meta}

\begin{tikzpicture}
  \coordinate (start) at (0, 0);
  \node[draw] (model) at (2, 0) {Model};
  \node[draw] (solve) at (4.5, 0) {Solve};
  \draw[-Stealth] (start) -- (model);
  \draw[-Stealth] (model) -- (solve);
  \draw[-Stealth] (solve.north) to [bend right]
    (model.north);
\end{tikzpicture}
```

Output



- `\coordinate` only names a point, here `start`.
- `\node` puts text at a place and optionally names it, here `model`.
- Names stand in for coordinates; arrows to a node stop at its border.
- `(model.north)` is the top of the node.

Styles and Positioning

Input

```
% in the preamble
\usepackage{tikz}
\usetikzlibrary{arrows.meta, positioning}

\begin{tikzpicture}[
    box/.style = {draw, rounded corners},
    > = Stealth,
  ]
  \node[box] (model) {Model};
  \node[box, right = of model] (solve) {Solve};
  \draw[->] (model) -- (solve);
\end{tikzpicture}
```

Output



- `box/.style` defines `box` as a style.
- `> = Stealth` sets the default tip for all arrows `->` in the picture to `Stealth`.
- `right = of model` needs the library `positioning`.

Loops: \foreach

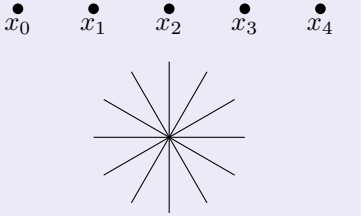
Input

```
\usepackage{tikz} % in the preamble

\begin{tikzpicture}
  \foreach \i in {0, ..., 4} {
    \fill (\i, 0) circle (2pt);
    \node[below] at (\i, 0) {$x_{\i}$};
  }
\end{tikzpicture}

\begin{tikzpicture}
  \foreach \a in {0, 30, ..., 330}
    \draw (0, 0) -- (\a:1);
\end{tikzpicture}
```

Output



- `\foreach` repeats a piece of the picture for each value in a list.
- `{0, 30, ..., 330}` continues in steps of the first two values.
- `(\a:1)` is the point at angle `\a` (in degrees) and distance 1.

Computed Coordinates: the Library calc

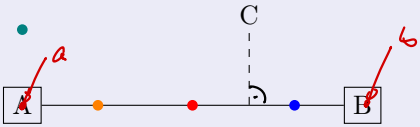
$(a)!\gamma!(b)$ $\hat{=}$ $(1-\gamma)(a) + \gamma(b)$

Input

```
% in the preamble
\usepackage{tikz}
\usetikzlibrary{calc}

\begin{tikzpicture}
  \node[draw] (a) at (0, 0) {A};
  \node[draw] (b) at (4.5, 0) {B};
  \node (c) at (3, 1.2) {C};
  \draw (a) -- (b);
  \fill[red] ($(a)!0.5!(b)$) circle (2pt);
  \fill[blue] ($(a)!0.8!(b)$) circle (2pt);
  \fill[orange] ($(a)!1cm!(b)$) circle (2pt);
  \fill[teal] ($(a) + (0, 1)$) circle (2pt);
  \draw[dashed] (c) -- ($(a)!(c)!(b)$);
\end{tikzpicture}
```

Output



($(a)!0.5!(b)$)	halfway from A to B
($(a)!0.8!(b)$)	80 % of the way
($(a)!1cm!(b)$)	1 cm from A toward B
($(a) + (0, 1)$)	1 above A
--($(a)!(c)!(b)$)	projection of C

Node names stand for the centers of the nodes.

Table of Contents

1. Graphics Files
2. TikZ Graphics
3. Best Practices
4. Errors and Obsolete Practices

When an AI Writes the Figure

- AI tools write Ti*k*Z code quickly.
- Whether the figure fits into the document is up to you.

Check generated Ti*k*Z code

- Compile it, and read the log: warnings point to problems that do not stop the run.
- Look for named styles instead of the same options repeated on every path, so that one change applies to all of them.
- Avoid special fonts and font sizes set in the picture, such as `\sffamily` or `\tiny`: labels should look like the text.
- Remove `\resizebox` around the picture, and libraries that are loaded but not used.
- Keep extensive code in a file of its own, so that you can change it later easily.

One File per Figure

Input: thesis.tex

```
\begin{figure}[htbp]
  \centering
  \input{figures/workflow.tex}
  \caption{From the model to its
    solution, as described in
    \zcref{section:methods}.}
  \label{figure:workflow}
\end{figure}
```

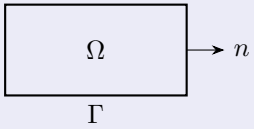
Input: figures/workflow.tex

```
\begin{tikzpicture}[
    box/.style = {draw, rounded corners},
    > = Stealth,
  ]
  \node[box] (model) {Model};
  ...
\end{tikzpicture}
```

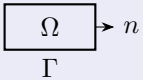
- `\input` reads the file at this place, as if its contents were written here.
- The figure file holds only the `tikzpicture`; the libraries go into the preamble of the main file.
- The float, its caption and its label stay in the main file, next to the text.

Making a Figure Smaller: `scale` and `\resizebox`

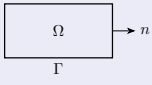
Original size



[scale = 0.5]



\resizebox



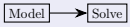
Original size



[scale = 0.5]



With `transform shape`



- `scale` shrinks the coordinates only: the text and the line widths keep their size, and nodes placed with `right = of` keep their relative position.
- With `transform shape`, `scale` shrinks the nodes as well, their text included.
- `\resizebox` (`graphicx.sty`) shrinks everything, the text too, which then no longer matches the font size of the document.

Shrinking the Drawing, Not the Text

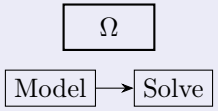
Input

```
% in the preamble
\usepackage{tikz}
\usetikzlibrary{arrows.meta, positioning}

\begin{tikzpicture}[scale = 0.5]
  \draw[thick] (0, 0) rectangle (2.4, 1.2);
  \node at (1.2, 0.6) {$\Omega$};
\end{tikzpicture}

\begin{tikzpicture}[node distance = 5mm]
  \node[draw] (model) {Model};
  \node[draw, right = of model] (solve) {Solve};
  \draw[-Stealth] (model) -- (solve);
\end{tikzpicture}
```

Output



Shrink the drawing, not the text

Use `scale` for coordinates and `node distance` for nodes placed with `right = of` (1 cm unless set). Avoid `transform shape` and `\resizebox`: the labels would no longer match the surrounding text.

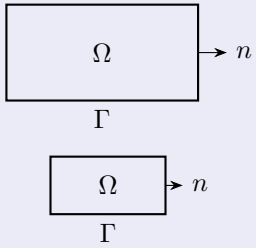
Drawing for the Width of the Text

Input

```
% in the preamble
\usetikzlibrary{arrows.meta}

\begin{tikzpicture}[x = 0.1\linewidth,
    y = 0.1\linewidth]
  \draw[thick] (0, 0) rectangle (4, 2);
  \node at (2, 1) {$\Omega$};
  \node[below] at (2, 0) {$\Gamma$};
  \draw[-Stealth] (4, 1) -- (4.6, 1)
    node[right] {$n$};
\end{tikzpicture}
```

Output for two \linewidth:s



Draw for the width the figure will have

Set the units x and y (1 cm by default) as a fraction of `\linewidth`. The drawing then fits the text and the labels stay in the font of the text.

Externalization: Compile Each Picture Only When It Changes

Input

```
% in the preamble
\usepackage{tikz}
\usetikzlibrary{external}
\tikzexternalize[prefix = figures/]
...
\tikzsetnextfilename{workflow}
\begin{tikzpicture}
  ...
\end{tikzpicture}
```

- The first run compiles each picture on its own, here into `figures/workflow.pdf`.
- Later runs include that PDF.
- When a picture changes, it is recompiled.
- `\tikzexternalize` needs `pdflatex -shell-escape`, and the folder `figures` must exist.

When and how to externalize

Externalize once the pictures make compiling slow, which may happen quickly with many or large pictures. Name each picture with `\tikzsetnextfilename{...}`: without a name, pictures are numbered, and inserting one recompiles all that follow.

Tantau, *The TikZ and PGF Packages*, 2025, pp. 670–673

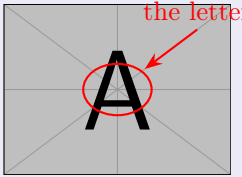
Annotating an Image

Input

```
% in the preamble
\usetikzlibrary{arrows.meta}

\begin{tikzpicture}
  \node[anchor = south west, inner sep = 0]
    (image) {\includegraphics[width = 3cm]
      {example-image-a}};
  \begin{scope}[x = {(image.south east)},
      y = {(image.north west)}]
    \draw[red, thick] (0.5, 0.5) circle (0.15);
    \draw[red, thick, -Stealth] (0.85, 0.85)
      node[above] {the letter} -- (0.62, 0.62);
  \end{scope}
\end{tikzpicture}
```

Output



- Given points, x and y become vectors: to the lower right and upper left corner of the image, which sits at the origin.
- $(0, 0)$ and $(1, 1)$ are then its lower left and upper right corner, at any image size.

Table of Contents

1. Graphics Files
2. TikZ Graphics
3. Best Practices
4. Errors and Obsolete Practices

Errors You Will Meet

Message	Usual cause
`! LaTeX Error: File `plot' not found.`	a typo in the file name, or the file in another folder
`! Package tikz Error: Giving up on this path. ...`	a path without its semicolon
`! Package pgfkeys Error: I do not know the key '/tikz/thik' ...`	a misspelled option
`! Package PGF Math Error: Unknown function `of' ...`	`right = of a` without the library `positioning`
`! Package tikz Error: You need to say \usetikzlibrary{calc} ...`	(`$...$`) without the library `calc`
`! Package pgf Error: No shape named `a' is known.`	an undefined node name, or a typo in a coordinate
`! Package tikz Error: Sorry, the system call '...' did NOT ...`	externalization without `-shell-escape`

Graphics Files: Mistakes and Obsolete Practices

Mistake without an error message	Effect
`width` and `height` together in `\includegraphics`	a distorted image (unless `keepaspectratio` is given)
a plot as a screenshot or JPG	blurred lines and letters when printed or enlarged
`trim` without `clip`	the trimmed parts are still drawn, over the text around the image
`\includegraphics{plot}` without file extension, with `plot.pdf` and `plot.png` both present	`plot.pdf` is used, even if `plot.png` is intended

Obsolete	Use instead	Why
`graphics.sty`	`graphicx.sty`	options by name, such as `width = 3cm`

TikZ: Mistakes and Obsolete Practices

Mistake without an error message	Effect
`\resizebox` or `transform shape` on a picture	labels in a font size other than that of the text
`scale` on a picture with labels	labels crowd the smaller drawing
`scale` on nodes placed with `right = of`	the diagram does not shrink
an annotated image not at the origin	marks in the wrong places
externalized pictures without `\tikzsetnextfilename`	inserting a picture recompiles all that follow

Obsolete	Use instead	Why
the library `arrows`	the library `arrows.meta`	declared deprecated in the TikZ manual
`above left = 1cm` without the library `positioning`	load the library `positioning`	declared deprecated in the TikZ manual; the library changes the result
`\tikzstyle{box} = [draw]`	`box/.style = {draw}`	no longer in the TikZ manual

Tantau, *The TikZ and PGF Packages*, 2025, pp. 190, 239

References

- Carlisle, D. P. (2025). *Packages in the 'graphics' Bundle*. The LaTeX Project. URL: https://ctan.org/pkg/graphics.
- Tantau, T. (2025). *The TikZ and PGF Packages. Manual for Version 3.1.11a*. Version 3.1.11a. URL: https://ctan.org/pkg/pgf.