Graphics for LaTeX users

Agostino De Marco

Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale

Gruppo Utilizzatori Italiani di TeX

GUIT

GUIT *meeting* *2019*

Politecnico di Torino
26 October 2019

Outline

Outline

General guidelines on illustration design

Drawing with LaTeX-aware software

- Using `Inkscape` + `TexText` extension

Drawing with natively available LaTeX environments/packages

- The standard environment `picture`
- The package `pstricks` (PostScript)
- The package `tikz`

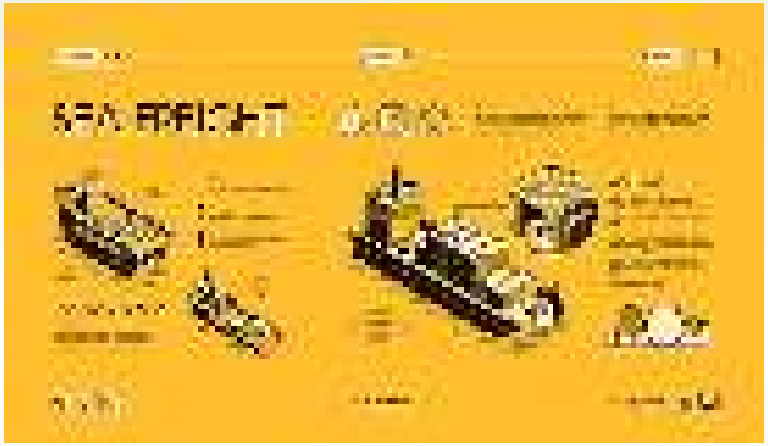
Data plots with package `pgfplots`

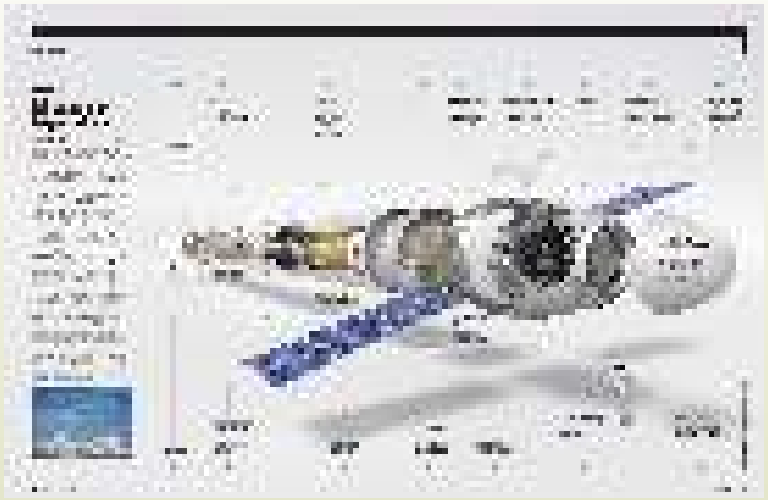
Illustrations

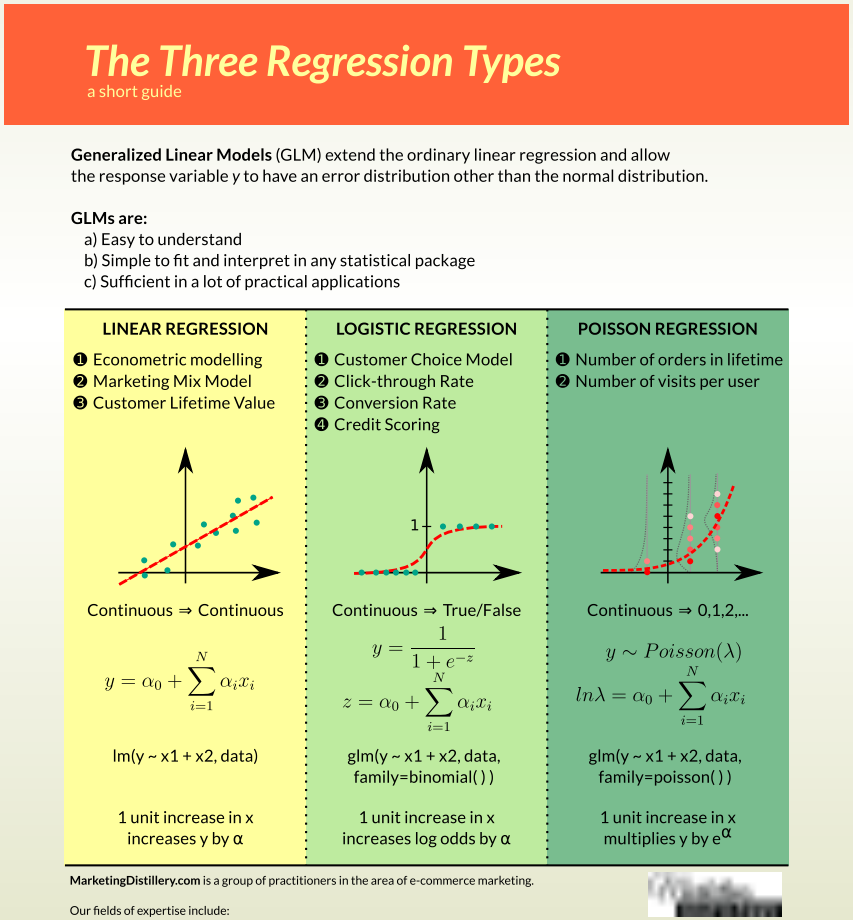
The term ***illustration*** refers to all kind of pictorial graphics — photographs, drawings, diagrams, and schematics.

Illustrations

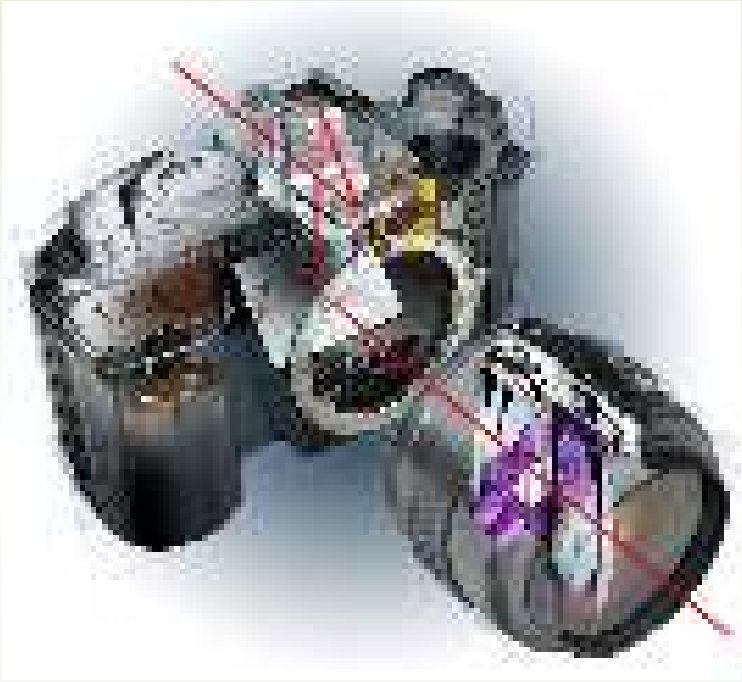
The term ***illustration*** refers to all kind of pictorial graphics — photographs, drawings, diagrams, and schematics.







Illustrations



(a) An example of technical illustration showing the Reflex principle.

(b) A newspaper illustration. This example shows a particular kind of artwork known as ‘infographics.’

Figure 1: Examples of on-the-job technical illustrations.

Illustrations

It is important in typography to ***maintain a consistency between text and graphics***.

Illustrations

Figure 2: A technical book in the hands of a reader. The right-hand page contains a full-height annotated illustration.

Illustrations – Benefits

Benefits coming from a careful use of ***visual material in technical documents***:

Illustrations – Benefits

Benefits coming from a careful use of ***visual material in technical documents***:

- ***Readers look for and want graphics***.

Illustrations – Benefits

Benefits coming from a careful use of ***visual material in technical documents***:

- ***Readers look for and want graphics***.
- Graphics enhance a communication's ***visual appeal***, thereby increasing the readers' concentration on its message.

Illustrations – Benefits

Benefits coming from a careful use of ***visual material in technical documents***:

- ***Readers look for and want graphics***.
- Graphics enhance a communication's ***visual appeal***, thereby increasing the readers' concentration on its message.
- ***Well-crafted graphics really can say more than many lines of text***, much more efficiently than prose.

Illustrations – Benefits

Benefits coming from a careful use of ***visual material in technical documents***:

- ***Readers look for and want graphics***.
- Graphics enhance a communication's ***visual appeal***, thereby increasing the readers' concentration on its message.
- ***Well-crafted graphics really can say more than many lines of text***, much more efficiently than prose.
- Graphics enable writers to ***convey information to readers who do not share a common language*** with the writers — or with each other.

Illustrations – Benefits

Benefits coming from a careful use of ***visual material in technical documents***:

- ***Readers look for and want graphics***.
- Graphics enhance a communication's ***visual appeal***, thereby increasing the readers' concentration on its message.
- ***Well-crafted graphics really can say more than many lines of text***, much more efficiently than prose.
- Graphics enable writers to ***convey information to readers who do not share a common language*** with the writers — or with each other.
- ***Graphics communicate information so effectively that they sometimes convey the entire message*** (see Figure 1a, Reflex camera).

Illustrations – Design guidelines

Keep in mind that, at some point, readers' attention will be going back and forth between text and figures, necessarily.

Illustrations – Design guidelines

Keep in mind that, at some point, readers' attention will be going back and forth between text and figures, necessarily.

Make the effort of having the readers feel at ease during the process.

Illustrations – Design guidelines

Keep in mind that, at some point, readers' attention will be going back and forth between text and figures, necessarily.

Make the effort of having the readers feel at ease during the process.

Design graphics with a special ***focus on usability***.

Illustrations – Design guidelines

Keep in mind that, at some point, readers' attention will be going back and forth between text and figures, necessarily.

Make the effort of having the readers feel at ease during the process.

Design graphics with a special ***focus on usability***.

Graphics should have the same good qualities of author's prose, ***easy for readers to understand and use***.

Engineering illustrations – Example

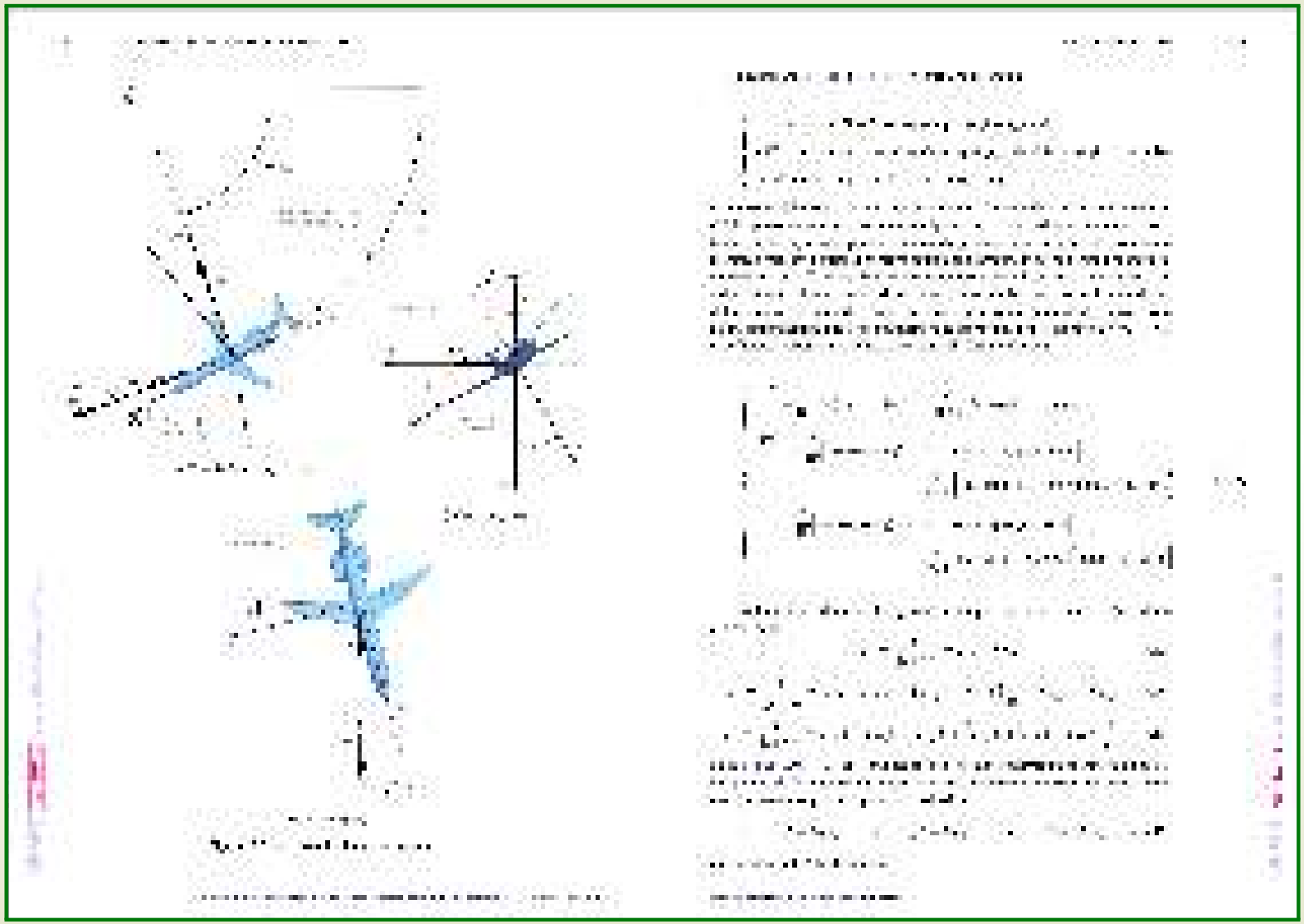


Figure 3: Aerospace engineering textbook.

http://wpage.unina.it/agodemar/DSV-DQV/DSV-DQV_Quaderno_17.pdf

Illustrations – Usability rules

Design to ***support any possible readers' tasks***. Imagine your readers in the act of using your graphic material.

Illustrations – Usability rules

Design to ***support any possible readers' tasks***. Imagine your readers in the act of using your graphic material.

Consider carefully your readers' knowledge and expectations. Specialized graphics as opposed to simplified visuals ('infographics').

Illustrations – Usability rules

Design to ***support any possible readers' tasks***. Imagine your readers in the act of using your graphic material.

Consider carefully your readers' knowledge and expectations. Specialized graphics as opposed to simplified visuals ('infographics').

Seek for simplicity. Especially for graphics that will be read on a computer screen or from a projected image. To keep it simple:

- Include only a manageable amount of material.
- Eliminate unnecessary details.

Illustrations – Usability rules

Design to ***support any possible readers' tasks***. Imagine your readers in the act of using your graphic material.

Consider carefully your readers' knowledge and expectations. Specialized graphics as opposed to simplified visuals ('infographics').

Seek for simplicity. Especially for graphics that will be read on a computer screen or from a projected image. To keep it simple:

- Include only a manageable amount of material.
- Eliminate unnecessary details.

Seek for the effectiveness of textual labels. Important content should always be labelled clearly.

Illustrations – Usability rules

Design to ***support any possible readers' tasks***. Imagine your readers in the act of using your graphic material.

Consider carefully your readers' knowledge and expectations. Specialized graphics as opposed to simplified visuals ('infographics').

Seek for simplicity. Especially for graphics that will be read on a computer screen or from a projected image. To keep it simple:

- Include only a manageable amount of material.
- Eliminate unnecessary details.

Seek for the effectiveness of textual labels. Important content should always be labelled clearly.

Choose effective informative titles (figure and table captions). Possibly, make them brief and informative at the same time.

Outline

Engineering illustrations – Example

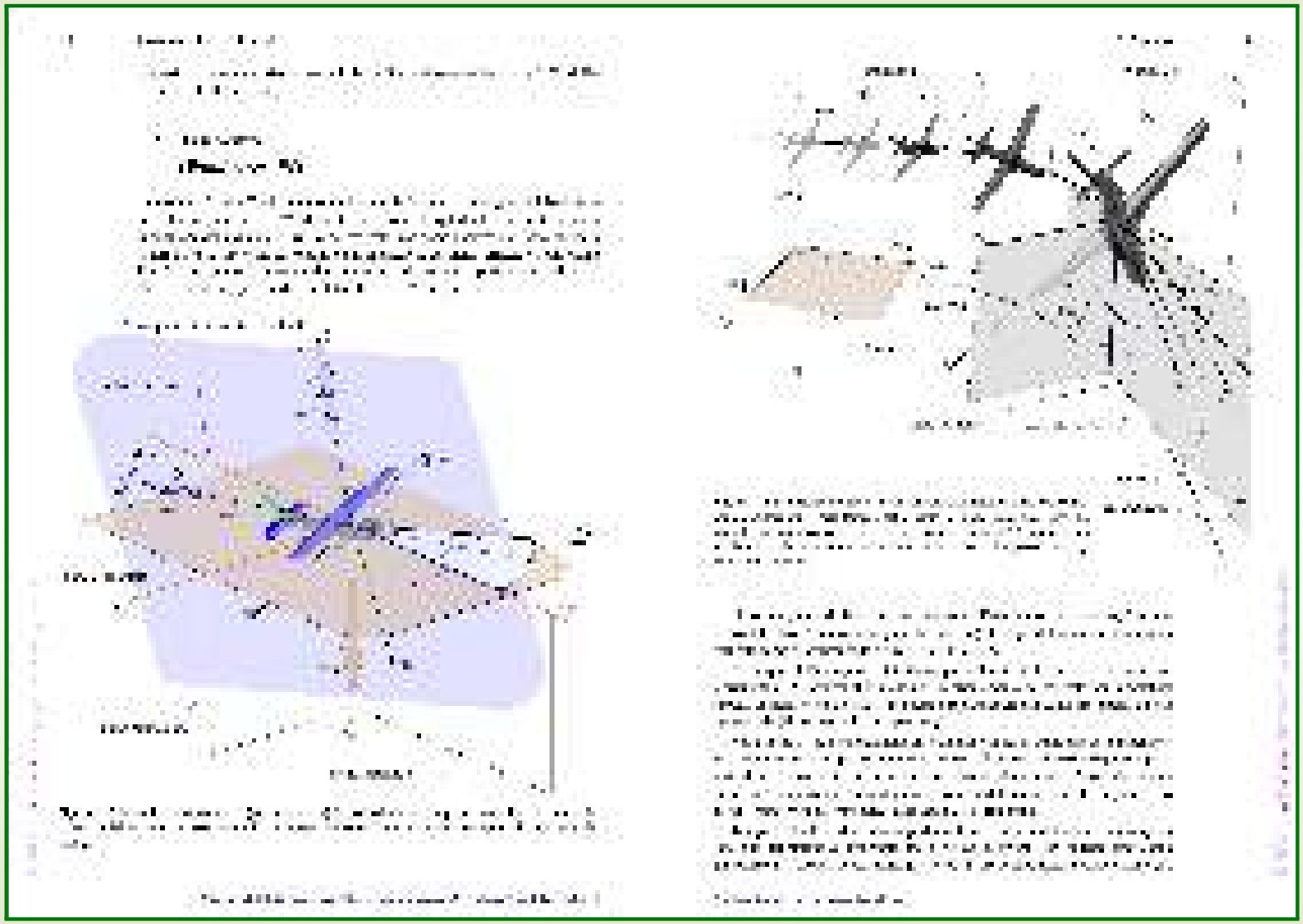


Figure 4: Aerospace engineering textbook.

http://wpage.unina.it/agodemar/DSV-DQV/DSV-DQV_Quaderno_1.pdf

Inkscape

`http://www.inkscape.org`

Inkscape is an ***open source*** and well-supported ***vector graphics/SVG editor*** available for all major operating systems.

Provides effective LaTeX-related capabilities, e. g. the TexText Python-based plugin extension.

`https://textext.github.io/textext`

TexText provides the possibility to ***add and re-edit (multi-line) LaTeX/XeLaTeX/LuaLaTeX generated SVG elements to a drawing***.

Inkscape

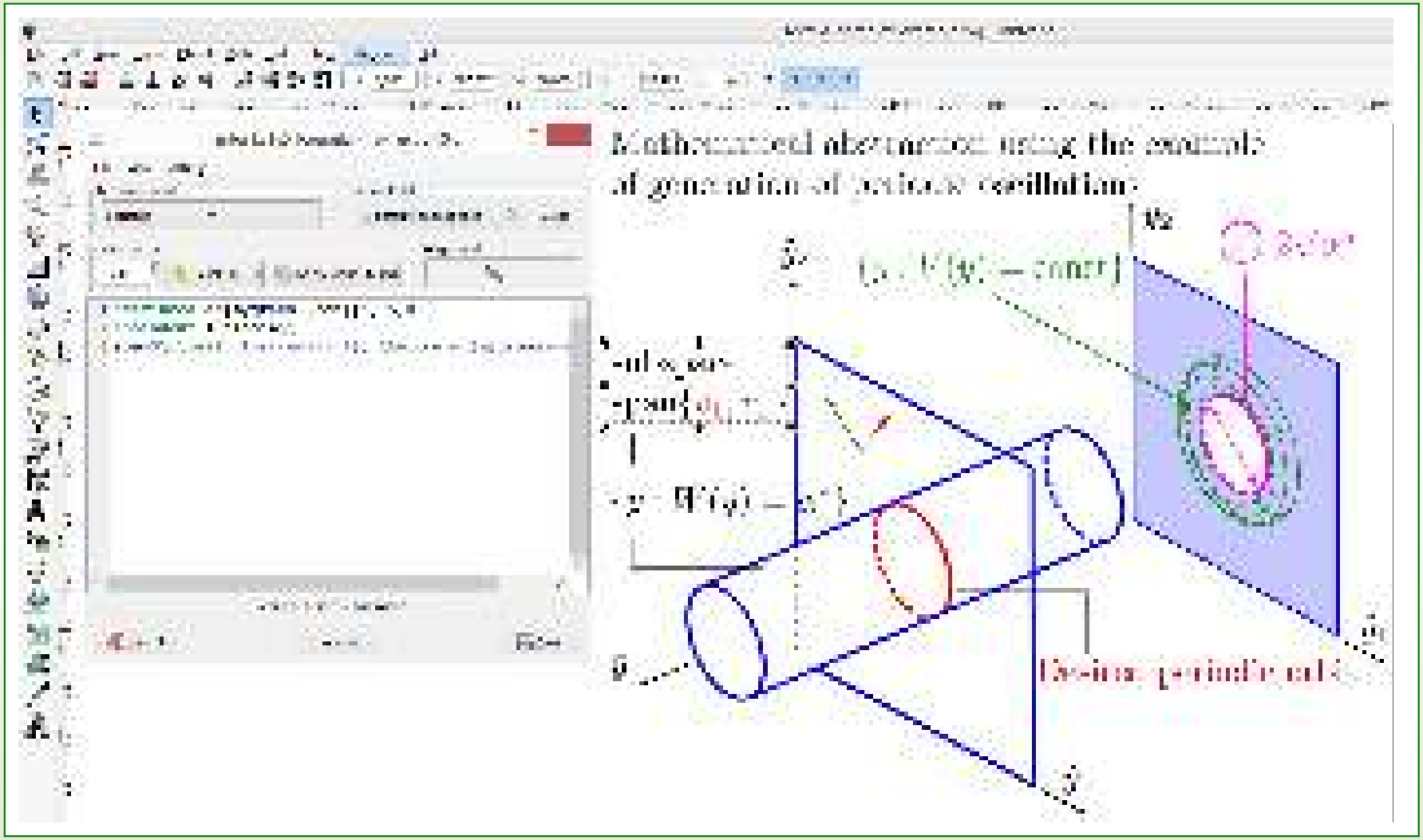
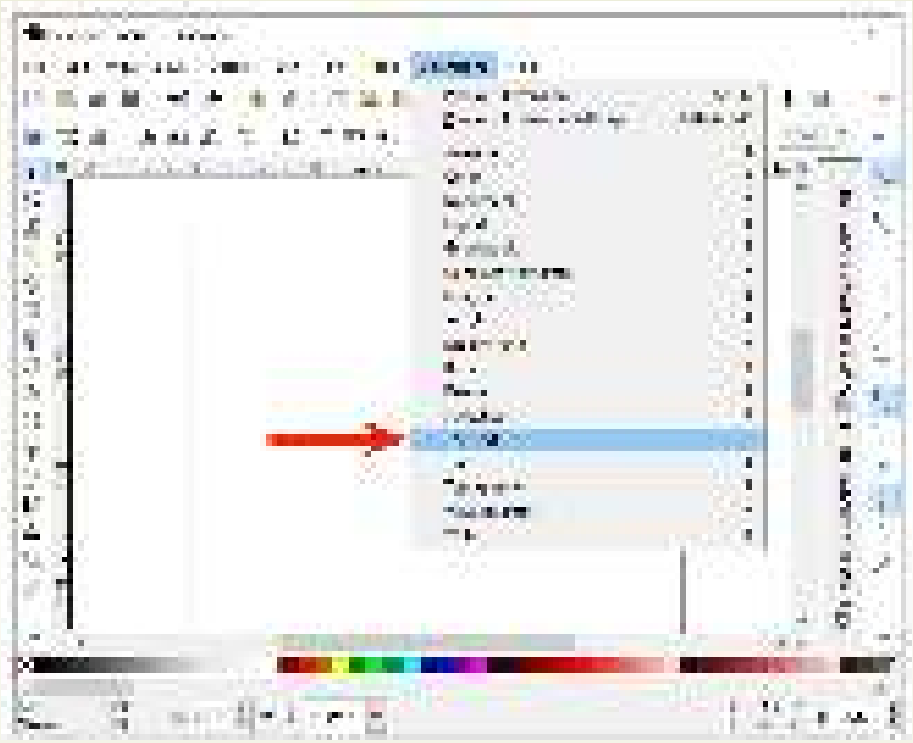


Figure 5: A screenshot of Inkscape with TexText extension in use.

Inkscape



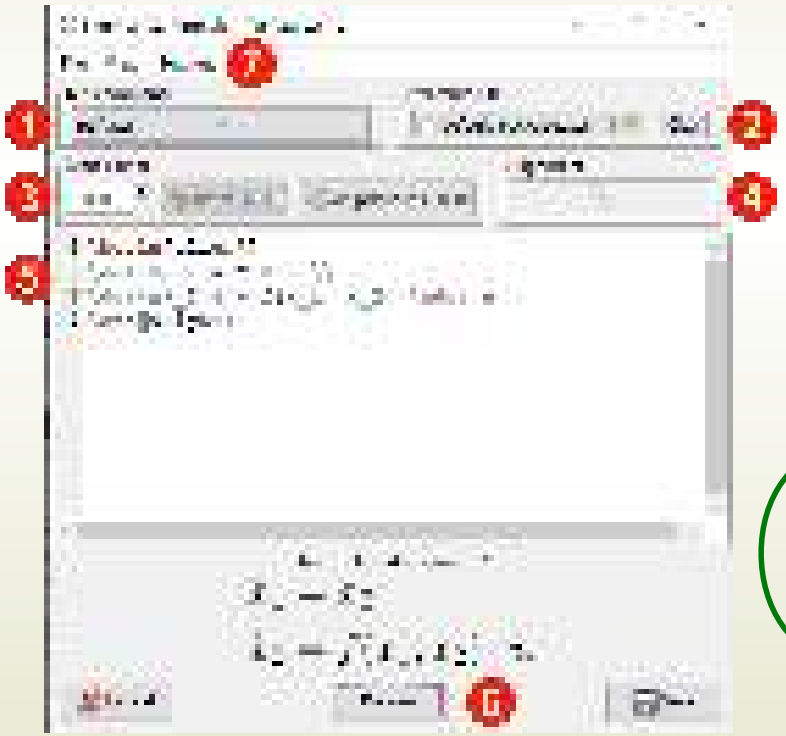
(a) Selecting TexText from Inkscape Extensions menu.

(b) The TexText dialog window.

Figure 6: Using TexText extension plugin in Inkscape.

Inkscape

```
% default_packages.tex
\usepackage{amsmath,amsthm,amssymb,
    amsfonts}
\usepackage{color}
```



2

customizable preamble

LaTeX template

```
\documentclass{article}
% ===> preamble file content <===
% default:
%   \input{default_packages}
\pagestyle{empty}
\begin{document}
% ==> User's code <===
\end{document}
```

5

user's content .dvi or .pdf → **SVG object**

Inkscape

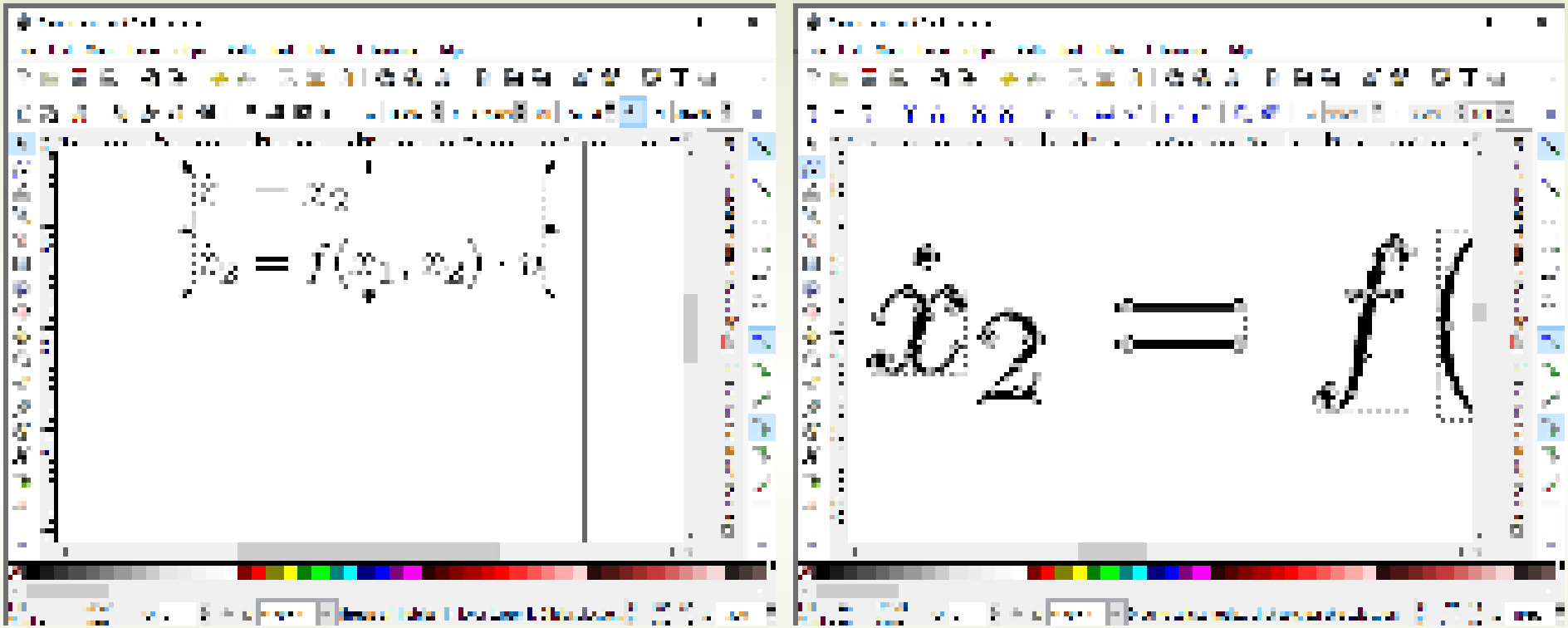


Figure 7: SVG element resulting from user's input compilation (see Figure 6b).

The final SVG object is re-editable via the* `TexText` *dialog!

Inkscape

Demo

Outline

Making drawings with code

A completely different paradigm.

No pseudo-synchronous visual tools.

Similar to the asynchronous typesetting workflow.

The native environment `picture`

```
% in preamble
\usepackage{pict2e}
% ...
\begin{picture}(120 ,80)
  \put(30,30){\circle*{3}}
  \put(30,33){\makebox(0,0)[br]{$A$}}
  \put(90,43){\circle*{3}}
  \put(88,47){\makebox(0,0)[bl]{$B$}}
  \linethickness{1.2pt}
  \Line(30,30)(90,43)
  \put(10,10){\vector(1,0){100}}
  \put(110,14){\makebox(0,0)[b]{$x$}}
  \put(10,10){\vector(0,1){60}}
  \put(14,70){\makebox(0,0)[l]{$y$}}
  % dashed box
  \put(0,0){\dashbox{5}(120,80){}}
\end{picture}
```

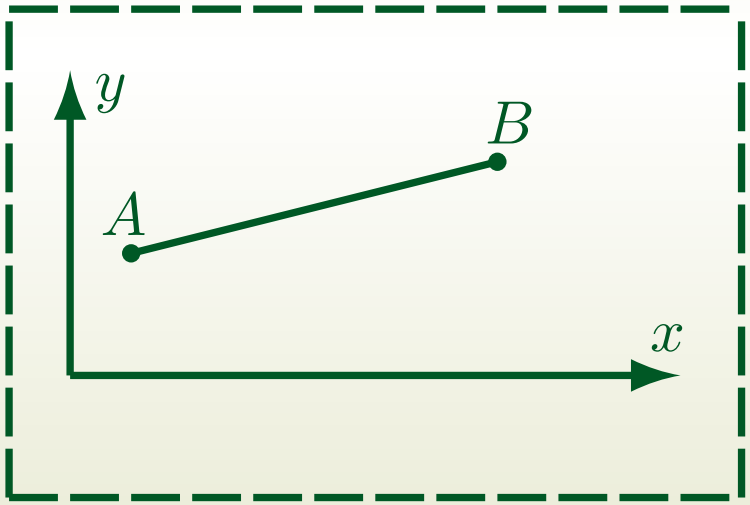


Figure 8: A drawing made with the standard `picture` environment enhanced by the `pict2e` package.

Drawing with pstricks

```
% arara: latex
% arara: dvips
% arara: ps2pdf
\documentclass[%
  border={0.6cm 0.6cm 0.6cm 0.6cm}% l b r t
  ]{standalone}
\usepackage[pdf]{pstricks}
\usepackage{pst-all}
\usepackage{pstricks-add}

\begin{document}
\begin{pspicture}(-1,0)(1,5)
  \psgrid[griddots=10,subgriddots=3,
    gridlabelcolor=blue](-1,0)(1,5)
  \psdots[linecolor=red,dotsize=10pt]
    (0,5)(-1,3)(1,2)(0.5,1)
  \rput(0,5){Center,Middle}
  \rput[bl](-1,3){%
    $\underbrace{\text{bottom,left}}_{\text{Really!}}$}
  \rput[Br](1,2){%
    $\underbrace{\text{Baseline,right}}_{\text{Really!}}$}
  \rput[tr]{45}(0.5,1){
    \parbox{5cm}{\flushright Rotated\\ by $45^{\circ}$}
  }
\end{pspicture}
\end{document}
```

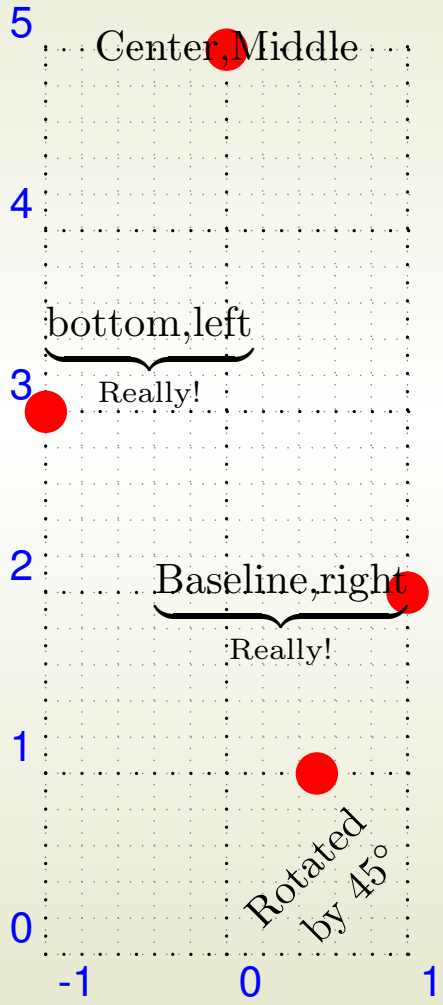


Figure 9: Placing whatever, wherever in a pspicture environment.

Drawing with packages pgf and tikz

Then whe have **pgf** and **tikz** by Till Tantau ...

https://www.ctan.org/pkg/pgf

http://texdoc.net/texmf-dist/doc/generic/pgf/pgfmanual.pdf

https://pgf-tikz.github.io/ (manual on the web)

Drawing with packages pgf and tikz

The name ***PGF*** means ***Portable Graphics Format***.

It is a package for creating ***inline graphics***: defines a number of TEX commands that can draw graphics within the typesetting process.

Graphics objects are put into boxes and treated as normal items to be taken care of by the LATEX output routine.

The package pgf exposes a ***frontend layer***, i. e. a set of commands or a special syntax that makes using the functionalities implemented by basic layer easier.

This frontend is what is called ***tikz***, the LATEX package tikz that incorporates pgf.

The name ***tikz*** is an acronym of ***Tikz Ist Kein Zeichenprogramm*** (German for 'tikz is not a drawing program').

Drawing with tikz

In preamble: `\usepackage{tikz}`

The package provides the command `\tikz` as in the following examples.

```
\tikz \draw (0pt,0pt) -- (20pt,6pt);
```

yields the line  , or

```
\tikz \fill[color=orange] (1ex,1ex) circle(1ex);
```

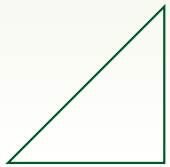
yields the orange circle  .

The argument passed to `\tikz` is a semicolon-terminated string.

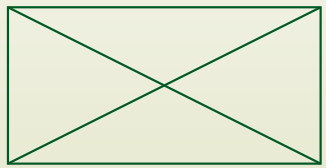
The tikzpicture environment

More elaborate drawings are embedded into the environment tikzpicture:

```
\begin{tikzpicture}
  \draw (0,0) -- (1,0) -- (1,1)
    -- cycle;
\end{tikzpicture}
```



```
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1);
  \draw (0,0) -- (2,1);
  \draw (0,1) -- (2,0.0);
\end{tikzpicture}
```



The `tikzpicture` environment

A `tikzpicture` can be used *inline* with the running text of a paragraph, like any other box object:

```
The following draws a
$0.4 \times 0.2$ crossed rectangle:
\begin{tikzpicture}
  \draw (0.0,0.0) rectangle
    (0.4,0.2);
  \draw (0.0,0.0) -- (0.4,0.2);
  \draw (0.0,0.2) -- (0.4,0.0);
\end{tikzpicture}\,.
```

The following draws a 0.4×0.2 crossed rectangle: ⧓.

Path extensions operations

Inside a `tikzpicture` environment ***everything is drawn by starting a path and by extending the path***. Paths are constructed using the `\path` command.

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
```

Path extensions operations

Inside a `tikzpicture` environment ***everything is drawn by starting a path and by extending the path***. Paths are constructed using the `\path` command.

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
```

directive or option

Path extensions operations

Inside a `tikzpicture` environment ***everything is drawn by starting a path and by extending the path***. Paths are constructed using the `\path` command.

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
```

directive or option

starting coordinate

Path extensions operations

Inside a `tikzpicture` environment ***everything is drawn by starting a path and by extending the path***. Paths are constructed using the `\path` command.

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
```

directive or option

starting coordinate

current coordinate

Path extensions operations

Inside a `tikzpicture` environment ***everything is drawn by starting a path and by extending the path***. Paths are constructed using the `\path` command.

```
\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
```

type of extension

directive or option

starting coordinate

current coordinate

Line-to and *move-to* operations

Command `\draw` stands for `\path[draw]`:

```
\path[draw] (0,0) -- (1,1);
\path[draw] (1,0) -- (2,0);
```

```
\draw (0,0) -- (1,1);
\draw (1,0) -- (2,0);
```

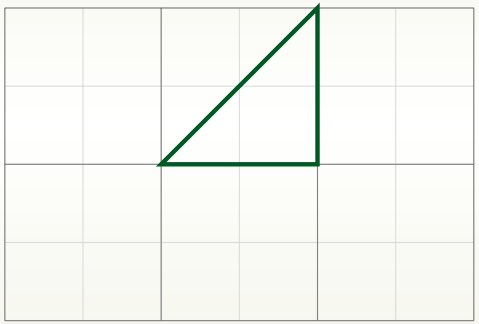
Multiple paths can be traced with one single command:

```
\draw (0,0) -- (1,1)
  % move-to operation
  (1,0) -- (2,0);
```

The *grid* and *cycle* operations

A **grid** is a path extension operation between two coordinates, much like a line (**--**):

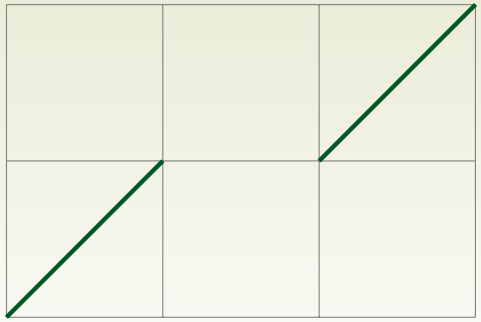
```
% fine, thin grid
\draw[line width=0.1pt,gray!30,step=5mm]
  (0,0) grid (3,2);
% coarse, thicker grid
\draw[help lines]
  (0,0) grid (3,2);
% a thick, closed path
\draw[thick] (1,1) -- (2,2) -- (2,1)
  -- cycle;
```



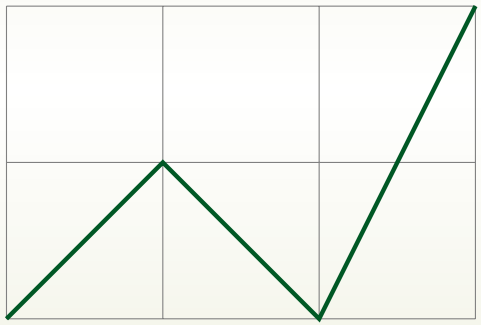
The **cycle** operation closes a path connecting the current point with the initial point on the path.

More *line-to* and *move-to* operations

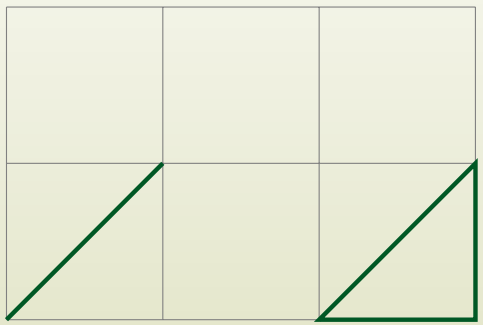
```
\draw[help lines] (0,0) grid (3,2);
\draw[thick] (0,0) -- (1,1)
  % then move-to
  (2,1) -- (3,2);
```



```
\draw[help lines] (0,0) grid (3,2);
\draw[thick] (0,0) -- (1,1) --
  (2,0) -- (3,2);
```

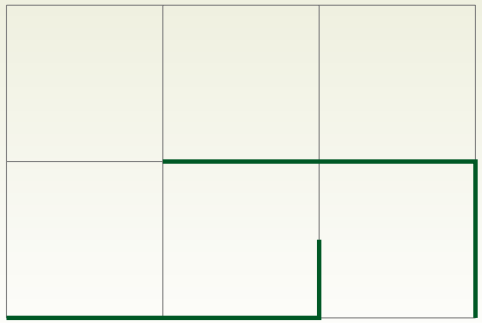


```
\draw[help lines] (0,0) grid (3,2);
\draw[thick] (0,0) -- (1,1)
  % then move-to
  (2,0) -- (3,0) --
  (3,1) -- cycle;
```

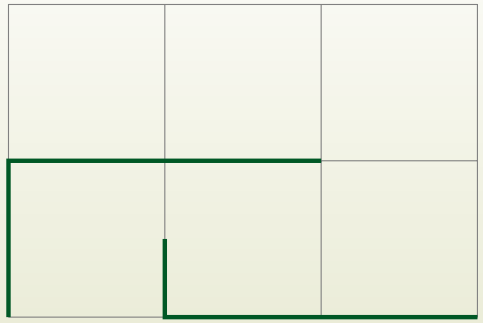


More *line-to* operations

```
\draw[thick] (0.0,0.0) -| (2.0,0.5)
  (1.0,1.0) -| (3.0,0.0);
```

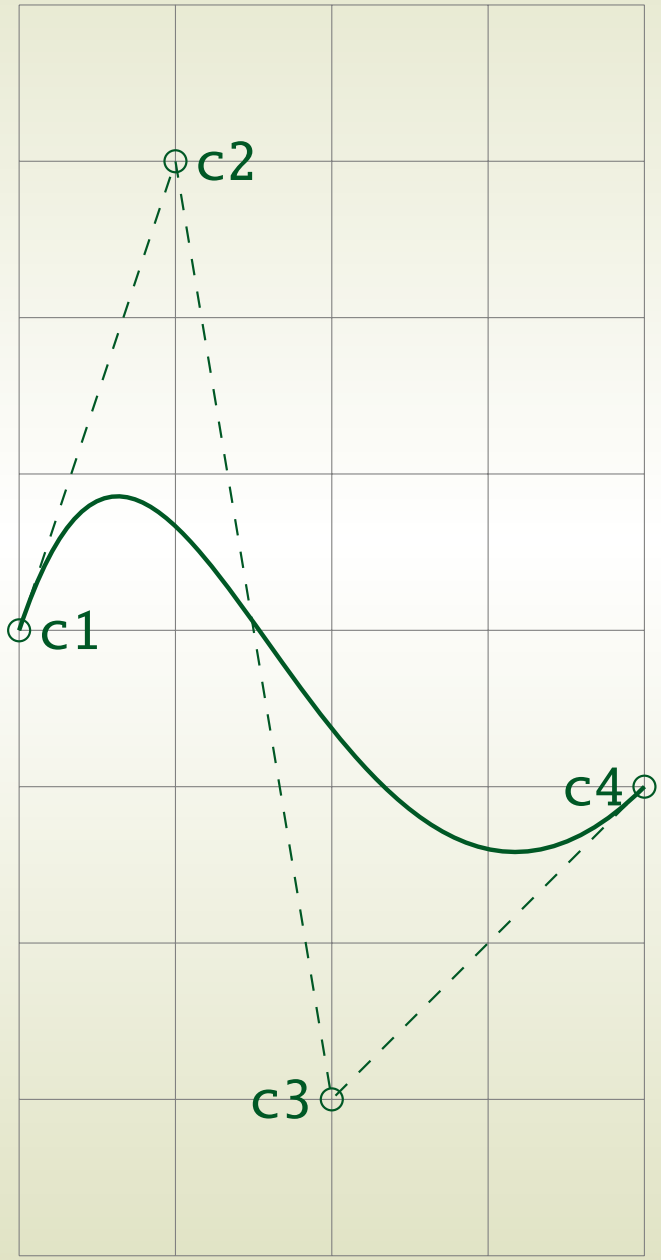


```
\draw[thick] (0.0,0.0) |- (2.0,1.0)
  (1.0,0.5) |- (3.0,0.0);
```



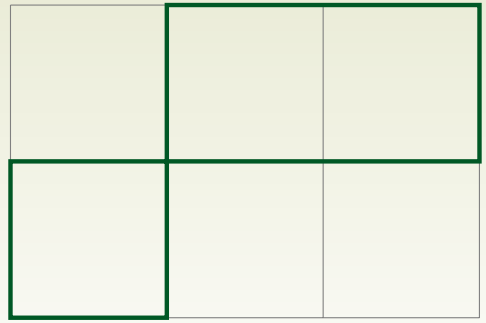
The *curve-to* operation

```
% the supporting grid
\draw[help lines] (-2,-4) grid (2,4);
% define labels (nodes)
\path (-2, 0) coordinate(c1)
  (-1, 3) coordinate(c2)
  ( 0,-3) coordinate(c3)
  ( 2,-1) coordinate(c4);
% segments connecting nodes
\draw[dashed] (c1) -- (c2) -- (c3) -- (c4);
% control points
\draw (c1) circle (2pt)
  (c2) circle (2pt)
  (c3) circle (2pt)
  (c4) circle (2pt);
% the Bézier curve
\draw[thick] (c1) .. controls (c2)
  \and (c3) .. (c4);
% text labels
\path
  (c1) node[anchor=west] {\texttt{c1}}
  (c2) node[anchor=west] {\texttt{c2}}
  (c3) node[anchor=east] {\texttt{c3}}
  (c4) node[anchor=east] {\texttt{c4}};
```

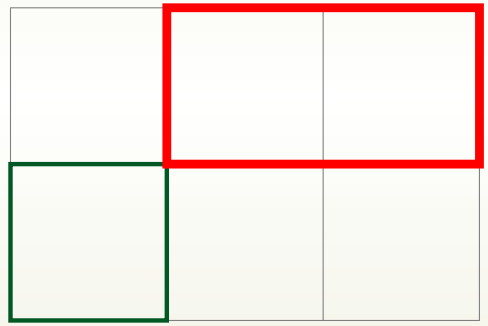


The *rectangle* and *circle* operations

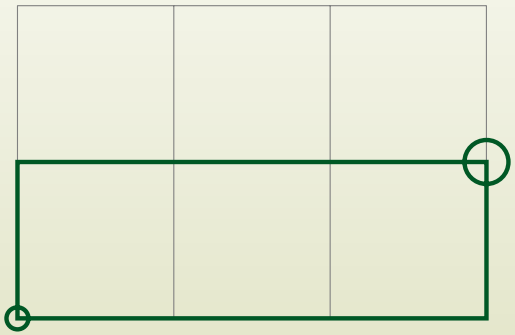
```
\draw[thick] (0,0) rectangle (1,1)
  rectangle (3,2);
```



```
\draw[thick] (0,0) rectangle (1,1);
\draw[ultra thick,red] (1,1)
  rectangle (3,2);
```

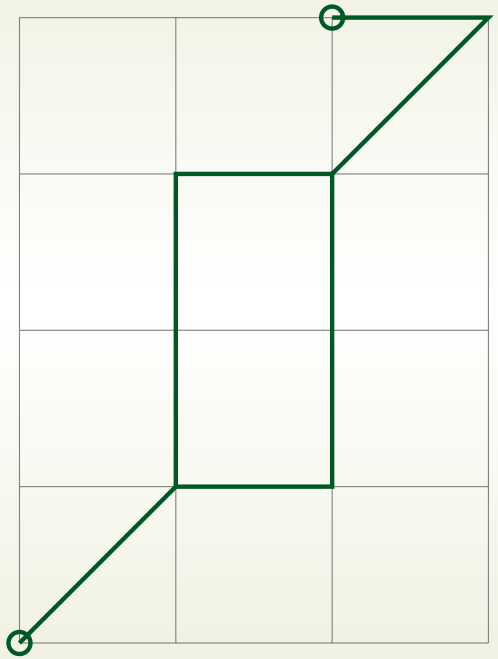


```
\draw[thick] (0,0) circle (2pt)
  rectangle (3,1)
  circle (4pt);
```



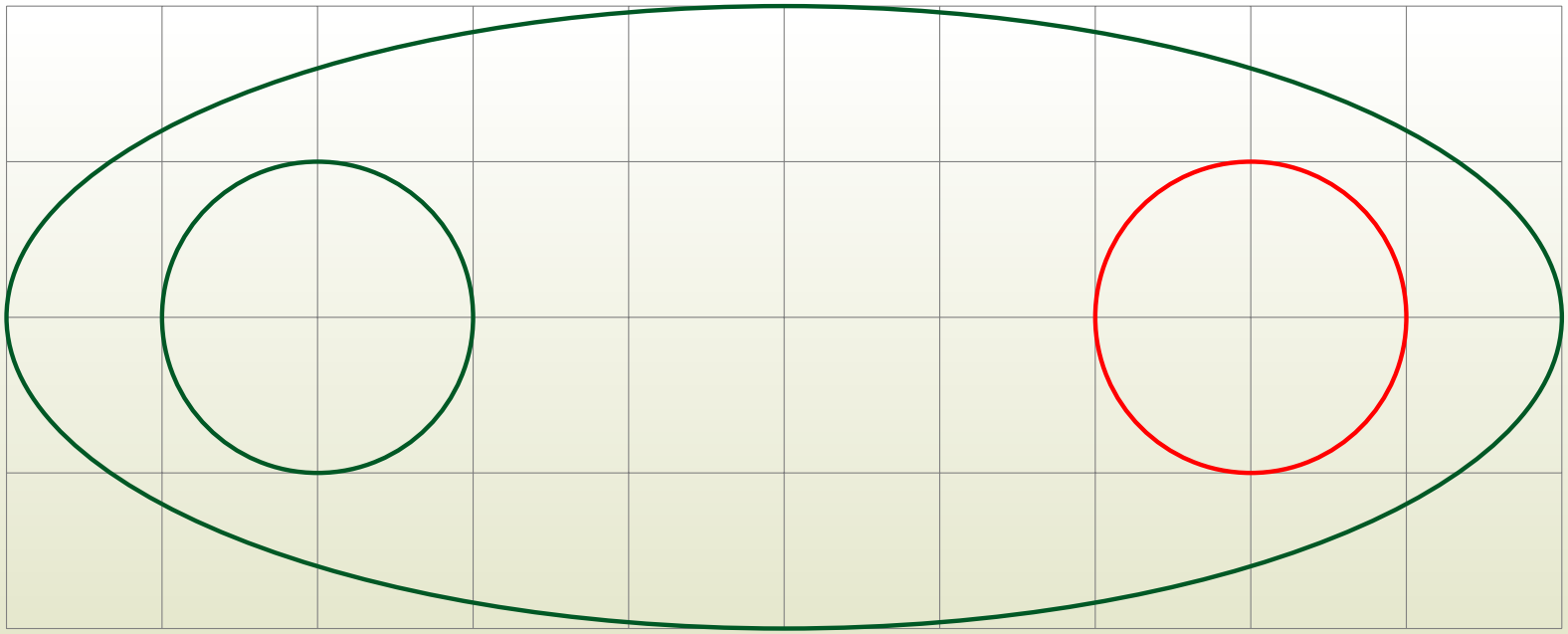
Multiple path extensions

```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (3,4);
  \draw[thick] (0,0) circle (2pt)
    -- (1,1) rectangle (2,3)
    -- (3,4)
    -- (2,4) circle (2pt);
\end{tikzpicture}
```



The *ellipse* operation

```
\draw[help lines] (0,0) grid (10,4);
\draw (2,2) ellipse (1cm and 1cm)
  (3,2) ellipse (3cm and 2cm);
\draw[red] (8,2) ellipse (1cm and 1cm);
```



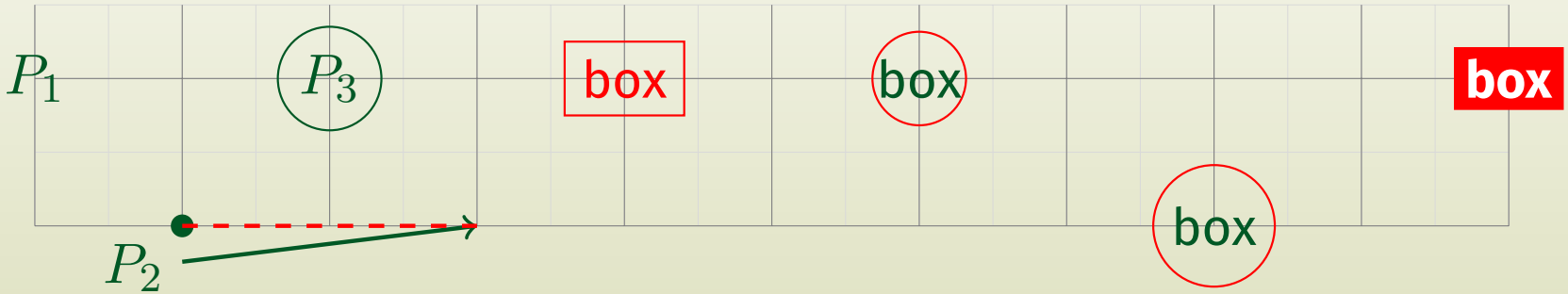
The *node* operation

You can add text, math, and other material to paths with the ***node path extension operation***.

The node operation

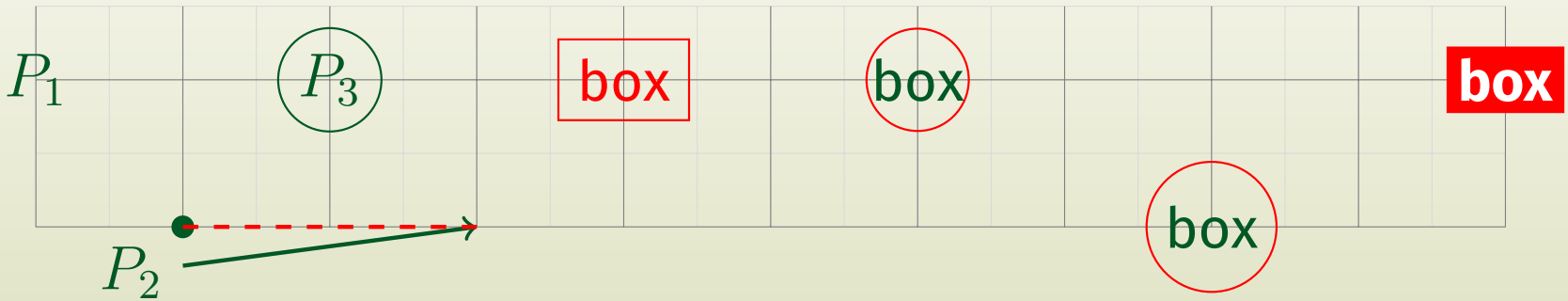
- places a given textual content at the current position;
- the current position becomes a node in the path;
- a label (variable name) can be associated to the node;
- named nodes can be used in further drawing operations;

Each node added to a path has an ***outer shape***. The outer shape is only drawn if `draw` is part of the options. ***The default node shape is a rectangle***.



The *node* operation

```
% in preamble: \usetikzlibrary{calc,positioning}
\path[draw] (0,1) node (p1) [draw=none] {$P_1$};
\path[draw,fill] (1,0) circle (2pt) node (p2) [anchor=north east] {$P_2$};
\path (2,1) coordinate (p3);
\path[draw] (p3) circle (10pt) node[draw=none] {$P_3$};
\path (3,0) coordinate (p4);
\draw[thick,->] (p2) -- (p4);
\draw[thick,dashed,red] (p2.north east) -- (p4);
\path (4,1) coordinate (p5);
\path[draw=none] (p5) circle (8pt) node[draw,red] {box};
\node [right=2.0cm of p5, anchor=center,
  inner sep=0pt, shape=circle, draw=red] (p6) {box};
\node[below right=1.0cm and 2.0cm of p6.center, anchor=center,
  inner sep=2pt, shape=circle, draw=red] (p7) {box};
\node [above right=1.0cm and 2.0cm of p7.center, anchor=center, inner
     sep=2pt, draw=red, fill=red, fill=white] (p7) {\textbf{box}};
```



Placing textual labels


```
\begin{tikzpicture}
  \draw (0,0)
    node (hello)
      [scale=2.0,
      inner sep=0pt,outer sep=0pt,
      draw=red]
      {\fbox{\textbf{Hello \GuIT}}};
  \draw (hello.north east) circle (2pt) node[anchor=south west] {north east};
  \draw (hello.north     ) circle (2pt) node[anchor=south     ] {north};
  \draw (hello.north west) circle (2pt) node[anchor=south east] {north west};
  \draw (hello.west      ) circle (2pt) node[anchor=east      ] {west};
  \draw (hello.south west) circle (2pt) node[anchor=north east] {south west};
  \draw (hello.south     ) circle (2pt) node[anchor=north     ] {south};
  \draw (hello.south east) circle (2pt) node[anchor=north west] {south east};
  \draw (hello.east      ) circle (2pt) node[anchor=west      ] {east};
\end{tikzpicture}
```

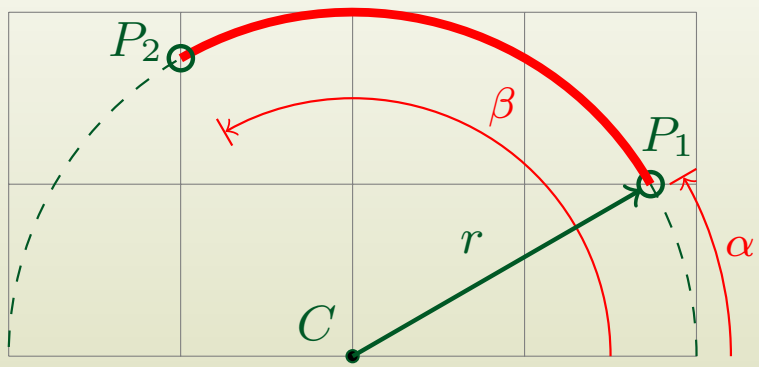
The *arc* operation

The arc operation adds an arc to the path.

- The arc starts at the current point, P_1. The user supplies two angles, α and β, and a radius r.
- The centre of the circle, C, is determined by the equation

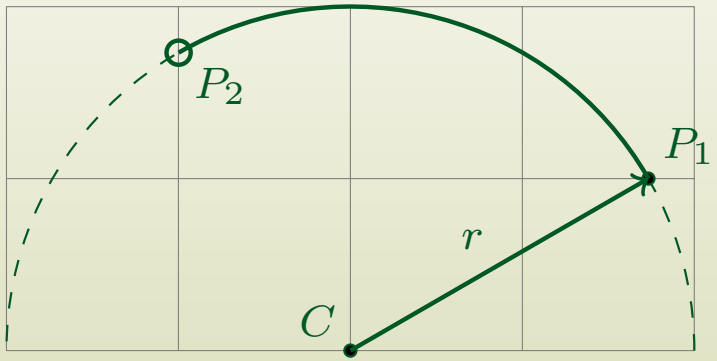
$$P_1 = C + (r\cos\alpha, r\sin\alpha)$$

  The end point of the arc is given by $P_2 = C + (r\cos\beta, r\sin\beta)$.
- The arc is drawn in counterclockwise direction from the start point to the end point, which becomes the new current coordinate of the path.



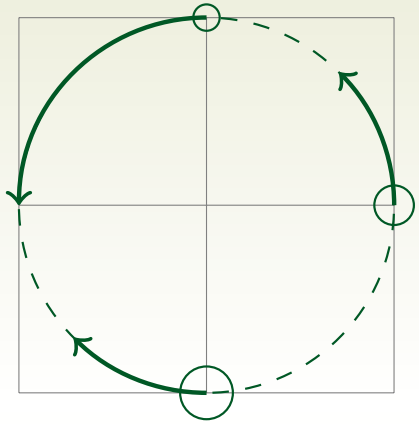
The *arc* operation

```
\draw[dashed] (4,0) coordinate (p0) arc (0:180:2cm); % (α, β, r)

\draw[fill=black] (2,0) coordinate (c) % ⟵ C
  circle (1pt) node[anchor=south east] {$C$};

\path (p0) arc (0:30:2cm) % (α, β, r), no arc drawn
  coordinate (p30); % ⟵ P_1

\draw[fill=black] (p30) circle (1pt) node[anchor=south west] {$P_1$};

\draw[thick] (p30) arc (30:120:2cm) % (α, β, r)
  coordinate (p120) % ⟵ P_2
  circle (2pt) node[anchor=north west] {$P_2$};

\draw[->,thick] (c) -- node[anchor=south east] {$r$} (p30); % ⟵ r⃗
```

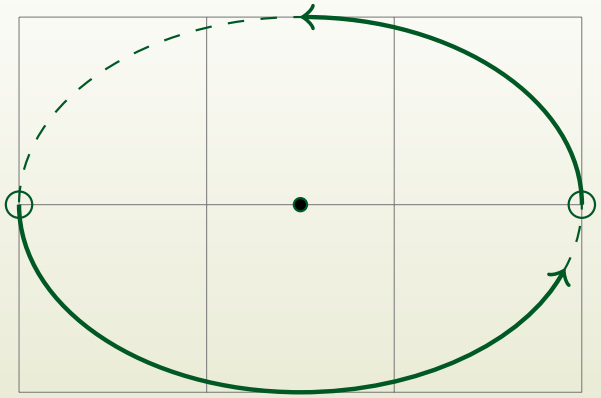


More *arc* operations

```
\draw[dashed] (1,1) circle (1cm);
\draw (1,2) coordinate (a) circle (2pt)
  (2,1) coordinate (b) circle (3pt)
  (1,0) coordinate (c) circle (4pt);
\draw[->,thick] (a) arc (90:180:1cm);
\draw[->,thick] (b) arc (0:45:1cm);
\draw[->,thick] (c) arc (270:225:1cm);
```



```
\draw[dashed] (1.5,0) circle (1.5cm and 1cm);
\draw[fill=black] (1.5,0) coordinate (c)
  circle (1pt);
\draw (3,0) coordinate (a) circle (2pt);
\draw (0,0) coordinate (b) circle (2pt);
\draw[->,thick] (a) arc (0:90:1.5cm and 1cm);
\draw[->,thick] (b) arc (180:340:1.5cm and 1cm);
```



Drawing with tikz

What else?

- More actions on paths, e.g. line widths, dash patterns, coloring, filling, shading.
- predefined styles of graphic elements and their customizations.
- Available coordinate systems and advanced coordinate calculations.

Please have a look at the article on ArsTEXnica for more details on tikz:

De Marco, A. "Graphics for LaTeX users".
ArsTEXnica 28 (October 2019), pp. 64–100.

All tikz examples given in the article are viewable on Overleaf:
https://www.overleaf.com/read/mgskyfdpttzt

Outline

Plotting data with pgfplots

The package **pgfplots** is built on top of pgf and is designed to draw graphs in a variety of formats, with a consistentand professional look and feel.

The package also allows to import data stored in files in tabular format via the package **pgfplotstable**.

As is usual with the pgf family, their manuals are impressive.

https://www.ctan.org/pkg/pgfplots
http://texdoc.net/texmf-dist/doc/latex/pgfplots/pgfplots.pdf

http://texdoc.net/texmf-dist/doc/latex/pgfplots/pgfplotstable.pdf

The axis environment

The workhorse of the pgfplots package is an environment called **axis**, which may ***define one or several plots*** (2D and 3D).

Each plot is drawn with the command **\addplot**.

The axis environment is used inside a tikzpicture environment.

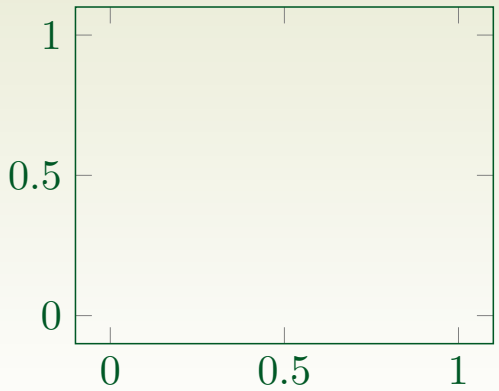
Typically, one or more plots are created in LaTeX as follows:

```
% in preamble
\usepackage{pgfplots}% loads tikz
...
\begin{tikzpicture}
   \begin{axis}[⟨graphic options⟩]
   ...
   ⟨pgfplots or tikz commands⟩
   ...
   \end{axis}
\end{tikzpicture}
```

The `axis` environment

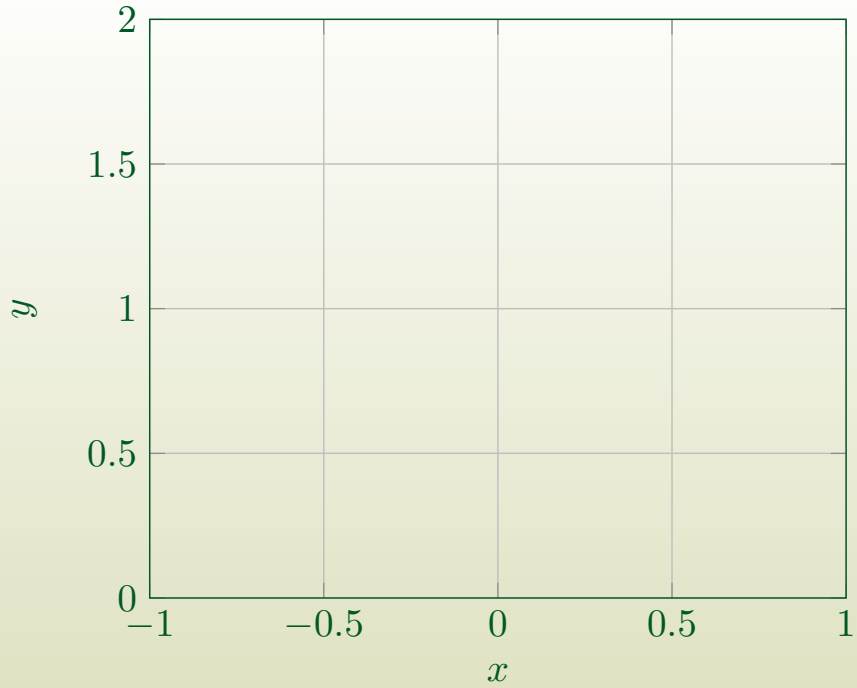
The simplest possible graph with `pgfplots`:

```
\begin{tikzpicture}
   \begin{axis}
   \end{axis}
\end{tikzpicture}
```



An empty `axis` environment, with customized formatting options:

```
\begin{axis}[
  xmin = -1, xmax = 1,
  ymin =  0, ymax = 2,
  grid = major,
  xlabel = $x$, ylabel = $y$
]
\end{axis}
```

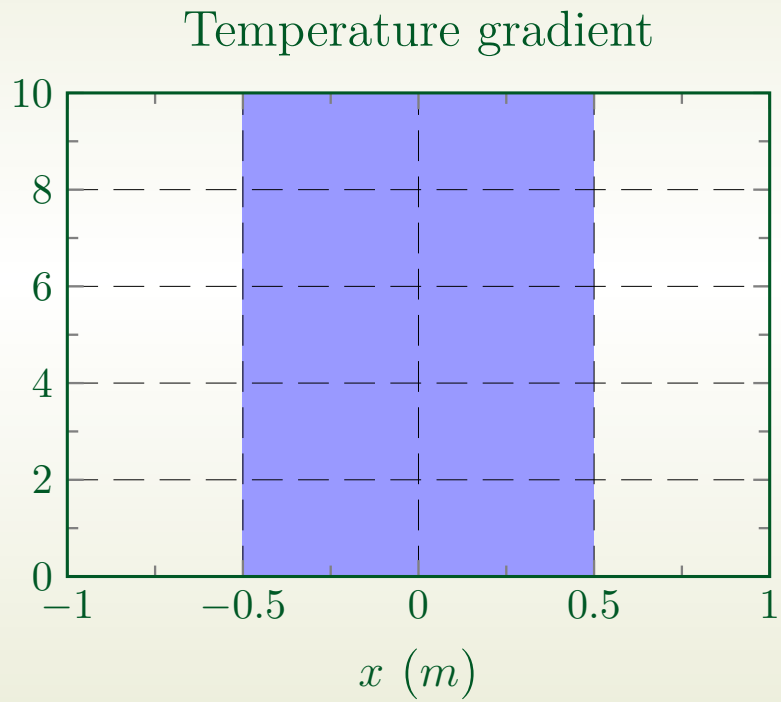


The axis environment

```
\begin{axis}[
  xmin = -1, xmax = 1,
  ymin = 0, ymax = 10,
  xtick = {-1,-0.5,...,1},
  ytick = {0,2,...,10},
  minor x tick num = 1,
  minor y tick num = 1,
  grid = major,
  xlabel ={$x$ (\si{\meter})},
  ylabel ={
    \parbox{2cm}{%
      \centering
      $\dfrac{\partial T}{\partial x}$
      \\[0.7em]
      \centering
      (\si{\celsius/\meter})
    }
  },
  title = {Temperature gradient},
  axis on top = true]
  % a basic tikz drawing command
  \fill[blue!40]
    (axis cs: -0.5, 0) -- (axis cs:  0.5, 0) --
    (axis cs:  0.5,10) -- (axis cs: -0.5,10) --
    cycle;
\end{axis}
```

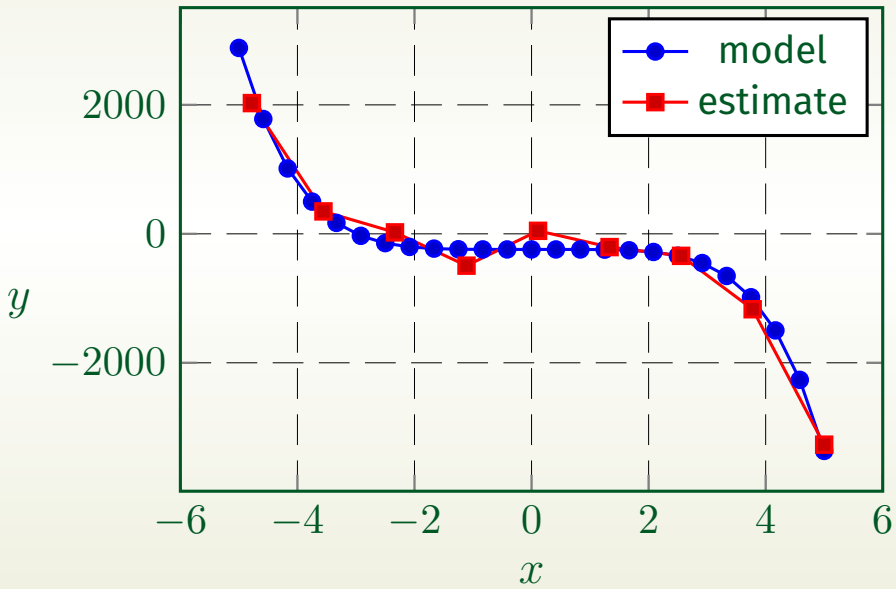
$$\frac{\partial T}{\partial x}$$

$$(°C/m)$$



The \addplot command

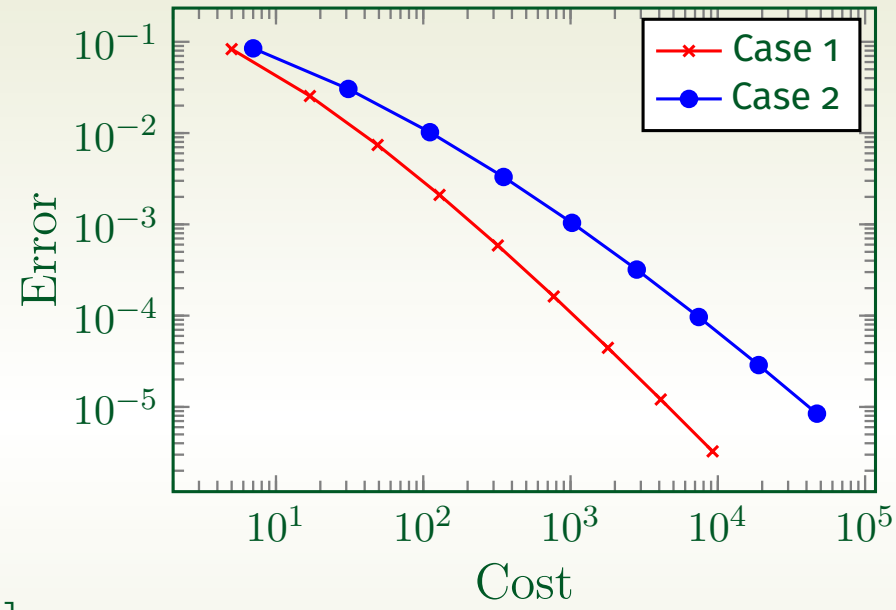
```
\begin{axis}[
  grid = major,
  xlabel = {$x$},
  ylabel = {$y$},
  y tick label style = {
    /pgf/number format/.cd,
    set thousands separator={}
  }]

  \addplot {-x^5 - 242};
  \addlegendentry{model}

  \addplot coordinates {
          (-4.77778,  2027.60977)
          (-3.55556,   347.84069)
          (-2.33333,    22.58953)
          (-1.11111,  -493.50066)
          ( 0.11111,    46.66082)
          ( 1.33333,  -205.56286)
          ( 2.55556,  -341.40638)
          ( 3.77778, -1169.24780)
          ( 5.00000, -3269.56775)
  };
  \addlegendentry{estimate}

\end{axis}
```



The \addplot command

Reading tabular data from file:

```
% in preamble
\usepackage{filecontents}
\begin{filecontents*}{data1.txt}
Level    Cost       Error
1           7 8.47178381e-02
2          31 3.04409349e-02
3         111 1.02214539e-02
4         351 3.30346265e-03
5        1023 1.03886535e-03
6        2815 3.19646457e-04
7        7423 9.65789766e-05
8       18943 2.87339125e-05
9       47103 8.43749881e-06
\end{filecontents*}
% ...
\begin{tikzpicture}
  \begin{loglogaxis}[xlabel={Cost}, ylabel={Error}]
  \addplot[color=red, mark=x] coordinates {
  (5,    8.31160034e-02)
  (17,   2.54685628e-02)
  (49,   7.40715288e-03)
  % ...
  (9217, 3.26101452e-06)
};
\addplot[color=blue, mark=*] table[x=Cost, y=Error] {data1.txt};
\legend{Case 1,Case 2}
\end{loglogaxis}
\end{tikzpicture}
```



Plotting data with pgfplots

What else?

- Data column manipulation with pgfplotstable.
- Style customizations of graphic elements.
- Available coordinate systems and advanced coordinate calculations. 3D plots.
- Exporting pgfplots sources from other data plotting tools.

Please have a look at the article on ArsTEXnica for more details on pgfplots:

- *De Marco, A., "Graphics for LaTeX users". ArsTEXnica 28 (October 2019), pp. 64–100.*
- *De Marco, A. and Giacomelli, R., "Creare grafici con pgfplots". ArsTEXnica 13 (October 2011), pp. 9–35.*

Conclusions

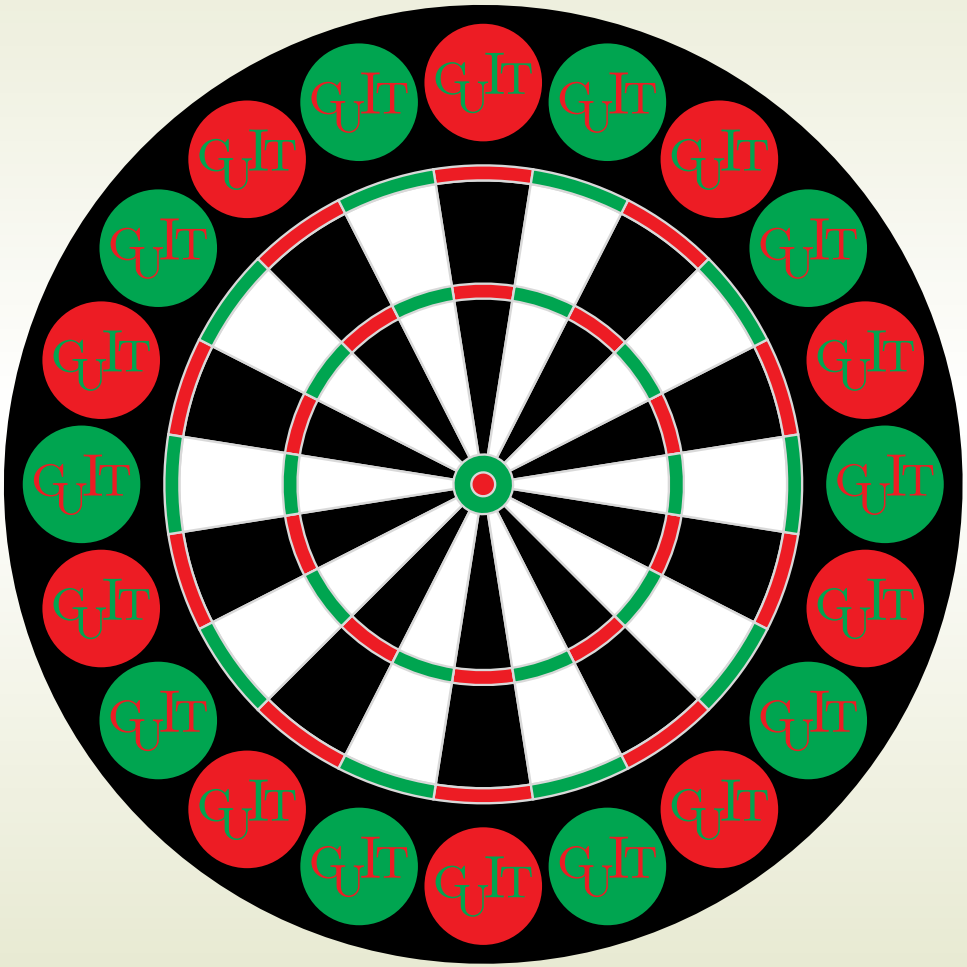
We have seen the most common scenarios encountered by LaTeX users when they face the problem of producing quality graphics.

In cases of diagrams, pictures and more or less complicated illustrations the two approaches based on package `tikz` and on the Inkscape graphics vector software have been presented.

Examples of scientific plots with the package `pgfplots`.

Thank you ...

Questions?



http://texample.net/tikz/examples/dartboard/