UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Lecture Notes

Werkzeuge für das wissenschaftliche Arbeiten

CS2450, Wintersemester 2015

Version November 16, 2015

Till Tantau

Contents

Vorwort

Liebe Studentinnen und Studenten,

in diesem Skript möchte ich Ihnen zwei Werkzeuge für das wissenschaftliche Arbeiten vorstellen, die ich selbst erschaffen / verbrochen habe: Beamer und Ti*k*Z. Die ersten Programmzeilen von Beamer sind vermutlich in etwa so alt wie Sie: Während meines Studiums habe ich mir ab vielleicht 1997 die ersten TEX-Makros geschrieben, mit denen ich Folien-Präsentationen einfacher erstellen konnte. Die heute recht bekannte Dokumentenklasse »Beamer« hat sich aus diesen Makros entwickelt, ihre Premiere unter diesem Namen hatte sie bei der Verteidigung meiner Dissertation 2013. Gleichzeitig war dies auch der erste Vortrag, den ich tatsächlich mit einem Beamer gehalten habe; damals eine sehr moderne (und nur sporadisch funktionierende) Technologie.

Die ersten Codezeilen zu Ti*k*Z sind etwas älter als Beamer, an Ti*k*Z begann ich während meiner Dissertation zu arbeiten. Der Anlass war, dass es damals schlichtweg kein Tool gab, mit dem man mit pdfLATEX Graphiken direkt im TEX-Quelltext erzeugen konnte. Da ich aber für meine Dissertation genau solche Graphiken brauchte und unbedingt pdfLATEX benutzen wollte, entstand Ti*k*Z.

Bei beiden Projekten war nicht abzusehen, dass aus »ein paar Makros für meine Dissertation« irgendwann umfängliche Pakete werden würden, die rund um den Globus vielfache Verwendung finden würden. Insbesondere hätte ich nie erwartet, dass beispielsweise das Ti*k*Z-Handbuch irgendwann die 1.000-Seiten-Marke sprengen würde.

Es wird mir nicht möglich sein, Ihnen in drei Doppelstunden der Inhalt von knapp 1.500 Seiten Manuals beizubringen. Glücklicherweise ist dies aber auch gar nicht nötig: Beide Werkzeuge sind so erworfen worden, dass es einfach ist, einfache Dinge damit zu tun und sich lediglich nach und nach an die komplexeren heranzuwagen.

Till Tantau

1-1

Chapter 1

Beamer: Strahlende Vorträge mit LaTeX

»That was the best PowerPoint presentation that I have ever seen!«

1-2

Chapter Objectives

1. Vorteile der Trennung von Inhalt und Form kennen
2. Benutzung von Beamer verstehen
3. Entscheiden können, wofür Beamer geeignet ist

Chapter Contents

1.1 Erste Schritte

1.1.1 Was ist das Beamer-Paket?

1-4

Mit dem Beamer-Paket erstellt man Präsentationen.

Was ist Beamer?

- Beamer erzeugt mit Hilfe von LaTeX aus *LaTeX-Manuskripten PDF-Präsentationen*.
- Beamer ist eine *Sammlung von LaTeX-Kommandos*.
- Beamer ist ein Open-Source-Projekt unter der *GNU Project License*.

Was ist Beamer nicht?

- Beamer ist *keine graphische Anwendung*.
- Beamer ist *kein* eigenständiges *Programm*.
- Beamer ist an *keine Plattform gebunden* (es läuft überall, wo TeX läuft).

1-5

Die Historie des Pakets.

1998 *Erste Codezeilen*, die noch heute genutzt werden.
Dies waren nützliche Makros während meines Studiums für erste Vorträge.

2003 *Feierliche erste Benutzung* des Pakets unter dem Namen *»Beamer«* im Rahmen meiner Promotionsverteidigung.

2003 Erster Upload von Version 0.10 auf *CTAN*. Zehn Bugreports in der ersten Woche.

bis 2007 Ständige Erweiterung des Pakets aufgrund von User-Anfragen.

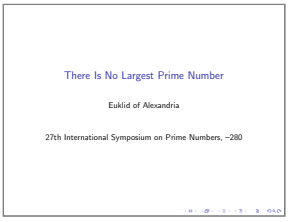
2007 Aktuelle *Version 3.07*.

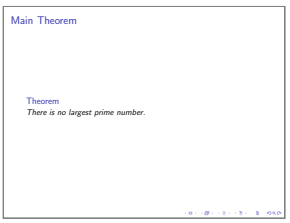
1.1.2 Vom Manuskript zum Vortrag

Vom Manuskript zum Vortrag mittels TeX. 1-6

Manuskript `presentation.tex` → TeX → Präsentation `presentation.pdf`

```
\documentclass{beamer}

\title{There Is No Largest Prime Number}
\author{Euklid of Alexandria}
\date[ISPN '80]{27th International
  Symposium on Prime Numbers, --280}

\begin{document}
  \maketitle

  \section{Results}

  \begin{frame}{Main Theorem}
    \begin{theorem}
      There is no largest prime number.
    \end{theorem}
  \end{frame}
\end{document}
```





Workflow: Vom Manuskript zum fertigen PDF. 1-7

1. Mit dem *Editor des eigenen Vertrauens* (wie Emacs oder LyX) erstellt man ein *Manuskript*.
   - Die LaTeX-Dokumentklasse muss `beamer` sein.
   - Man benutzt Standard-LaTeX-Befehle zur Strukturierung wie `\section` oder beispielsweise `{itemize}`.
   - Man benutzt spezielle Beamer-Befehle für Sondereffekte wie `{frame}` oder auch `\only<5>`.
2. Man *übersetzt* das Manuskript mit *LaTeX*.
3. Man *prüft* das resultierende PDF.
4. Man beginnt wieder von vorne, indem man das Manuskript korrigiert.

Wichtige Fähigkeiten des Beamer-Pakets. 1-8

- Automatisches Inhaltsverzeichnis.
- Automatische Navigationsleisten.
- Einfaches Ein- und Ausblenden von Seitenteilen.
- Unterstützung nichtlinearer Vorträge.
- Gute Voreinstellungen für das Aussehen von Vorträgen.
- Konfigurierbares Aussehen von Vorträgen.
- Erzeugung von Textfassungen aus dem Manuskript.

1.1.3 Trennung von Inhalt und Form

Wie gedruckte Texte entstehen. 1-9

Früher

- *Autoren* schreiben den *Inhalt* eines Textes auf.
- *Typographen* bestimmen das *allgemeine Aussehen* des Textes.
- *Drucker* verteilen den Inhalt über die Seiten hinweg.

Heute

- *Autoren* schreiben den *Inhalt* in Textverarbeitungen auf.
- *Autoren* bestimmen das *Aussehen* des Textes.
- *Programme* verteilen den Inhalt über die Seiten hinweg.

Leider haben Autoren in der Regel von Typographie keine Ahnung.

1-10

Wiederholung: Eine Grundphilosophie für erfreuliche Drucksachen.

»Trenne Inhalt, Struktur und Form.«

- Der *Inhalt* ist der »reine Text«.
- Die *Struktur* gibt an, wie der Text aufgebaut ist.
- Die *Form* gibt an, wie die ganze Sache aussieht.

1-11

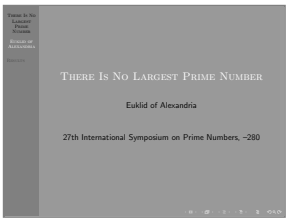
Wiederholung: Vorteile der Trennung von Inhalt und Form bei Beamer.

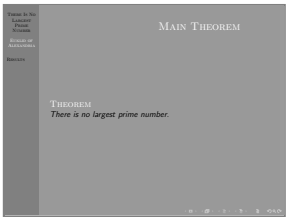
1. Die Form kann *neuen Bedingungen* angepasst werden.
   - Eine xml-Datei kann auf einem *Monitor*, auf einem *Ausdruck* und auf einem *Handheld unterschiedlich aussehen*.
   - Eine tex-Datei kann in einer *Vorlesungspräsentation* ganz anders aussehen als in einem *gedruckten Skript*.
2. Die Form ist *einheitlich*.
   - Schriften, Größen und Farben sind automatisch einheitlich.
   - Werden Form und Inhalt vermischt, so gelingt dies oft nicht (beispielsweise manchmal bei PowerPoint).

1-12

Gleicher Inhalt, andere Formen.

```
\documentclass{beamer}

\usetheme{Hannover}
\usecolortheme{fly}
\usefonttheme{structuresmallcapsserif}

\title{There Is No Largest Prime Number}
\author{Euklid of Alexandria}
\date[ISPN '80]{27th International
  Symposium on Prime Numbers, --280}

\begin{document}
  \maketitle

  \section{Results}

  \begin{frame}{Main Theorem}
    \begin{theorem}
      There is no largest prime number.
    \end{theorem}
  \end{frame}
\end{document}
```

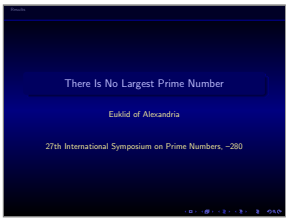




1-13

Gleicher Inhalt, andere Formen.

```
\documentclass{beamer}

\usetheme{Frankfurt}
\usecolortheme[overlystylish]{albatross}

\title{There Is No Largest Prime Number}
\author{Euklid of Alexandria}
\date[ISPN '80]{27th International
  Symposium on Prime Numbers, --280}

\begin{document}
  \maketitle

  \section{Results}

  \begin{frame}{Main Theorem}
    \begin{theorem}
      There is no largest prime number.
    \end{theorem}
  \end{frame}
\end{document}
```



1.2 Fallbeispiele

1.2.1 Ein mathematischer Vortrag

Zeigen der Folien des Vortrags zum Thema On the Complexity of Kings, gehalten auf der FCT 2007 Tagung. Regie

Überblick zu einem typischen mathematischen Konferenzvortrag. 1-14

- Der Vortrag wurde auf einer Theorietagung gehalten.
- Das gewählte *Theme lässt viel Platz* für den Inhalt, die Navigation ist nicht aufdringlich.
- Die *Kontraste sind maximal*, da die Bedingungen vor Ort unbekannt waren.
- Die Schrift ist *recht fett* und auch bei schlechter Auflösung gut lesbar.
- Es wird viel mit *Graphiken* und *Overlays* gearbeitet.

1.2.2 Eine Vorlesungsreihe mit Skript

Zeigen der Folien und des Skriptes zu dieser Vorlesung. Regie

Überblick zu einem Vorlesungsskript. 1-15

- Die Vorlesung besteht aus vielen Kapiteln, jedes ist eine eigene Präsentation.
- Das Skript wird aus *derselben Quelle* wie die Präsentation erzeugt.
- Die Skriptvariante ist aber *platzsparender* und *für kontinuierliches Lesen* geeignet.
- In der Präsentationsversion ist die *Navigation größer*, da Vorlesungen viel länger dauern als Konferenzvorträge.

1.3 Stärken und Schwächen

Beamer versus graphische Präsentationstools. 1-16

	Beamer	graphische Tools
Erlernen ohne LaTeX-Kenntnisse	⊖⊖	⊕
Objekte frei positionieren	⊖	⊕⊕
Graphiken direkt erstellen	⊖	⊕
Einbinden von Multimedia	○	⊕
Arbeitsgeschwindigkeit Anfänger	○	○
Arbeitsgeschwindigkeit Profi	⊕	⊕
Erlernen mit LaTeX-Kenntnissen	⊕	⊕
Dokumentation	⊕	⊕
Vorlagenqualität	⊕	○
Typographie	⊕	⊖⊖
Konsistenz des Aussehens	⊕⊕	⊖
Visualisierung des Vortragsaufbau	⊕⊕	⊖
Mathematische Formeln	⊕⊕	⊖⊖
Quelltextanzeige	⊕⊕	⊖⊖

Chapter Summary

1. *Beamer* ist eine *LaTeX-Kommandosammlung* zur Erstellung von Präsentationen. 1-17
2. Die *Trennung von Form und Inhalt* erschwert Anfängern die Benutzung, hat aber *Vorteile*:
   - leichtere Wiederverwendbarkeit,
   - konsistentes, professionelles Aussehen.
3. *Stärken* sind
   - mathematischer Formelsatz,
   - automatische Navigationsleisten,
   - robuste Standardvorlagen,
   - ausführliche Dokumentation,
   - die Erstellung von speziellen Textfassungen.

2-1

Chapter 2
An Overview of TikZ
A Language for Creating Graphics the TEX Way

2-2

Chapter Objectives

1. You can name and use the basic primitives of TikZ to create different kinds of graphics.
2. You can describe the system design underlying TikZ.
3. You can describe how TikZ works on an abstract level.

Chapter Contents

2.1 Goal-Oriented Overview – Creating a Figure

2.1.1 How Do I Use TikZ?

2-4

What Is TikZ?

- "TikZ ist *kein* Zeichenprogramm." (TikZ is not a drawing program.)
- TikZ is a TEX macro package.
- Just as TEX provides a special notation for formulas, TikZ provides a special notation for graphics.

2-5

Formulas In TEX – Graphics in TikZ

In TEX you write

```
Let $\int_0^1
\sqrt{x}\, dx$
be the integral, \dots
```

and get

Let $\int_0^1 \sqrt{x}\,dx$ be the integral, …

In TikZ you write

```
See \tikz \draw[->]
(0,0)-(2ex,1ex);
here \dots
```

and get

See ↗ here …

Installation and Usage of the Package. 2-6

1. The package is preinstalled in all major distributions.
2. Add to your documents:

```
\usepackage{tikz}           % For LaTeX
\usetikzlibrary{arrows.meta,petri,...}

\input tikz.tex             % For plain TeX
\usetikzlibrary{arrows.meta,petri,...}

\usemodule[tikz]            % For ConTeXt
\usetikzlibrary[arrows.meta,petri,...]
```

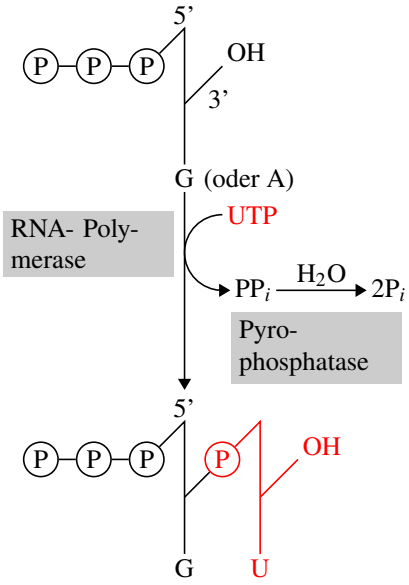
3. Process the file using one of the following:
   - `pdf(la)tex`
   - `(la)tex` and `dvips`
   - `(la)tex` and `dvipdfm`
   - `xe(la)tex` and `xdvipdfmx`
   - `vtex`
   - `textures`
   - `tex4ht`

History 2-7

- The PGF system underlying Ti*k*Z was created for the graphics in my PhD thesis.
- The first lines of code were written around 2000.
- The manual that comes with the package is around 1.200 pages and *very* detailed.

2.1.2 Recreating a Figure From a Biochemistry Textbook

Our Goal: Recreating This Figure. 2-8



Our aim is to create this figure using Ti*k*Z.

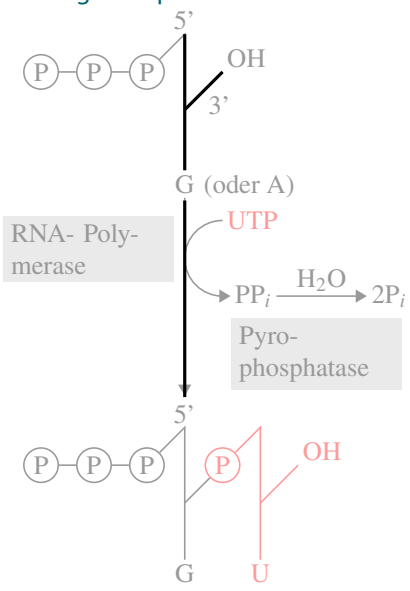
The figure is a redrawing of the figure on page 128 of the text book

References

[1] Chirsten Jaussi Biochemie Springer-Verlag, 2005

2-9

Drawing a Simple Line.

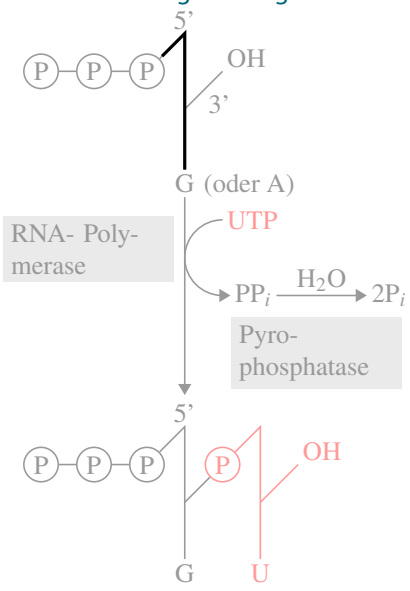



```
\begin{tikzpicture}
  \draw (5mm,59mm) -- (5mm,41mm);
  \draw (5mm,49mm) -- (10mm,54mm);
  \draw (5mm,37mm) -- (5mm,11mm);
   ...
\end{tikzpicture}
```

- TikZ-commands have to be given in a `{tikzpicture}` environment.
- The picture size is calculated *automatically*.
- First command: `\draw`.

2-10

A Path Consisting of Straight Lines.

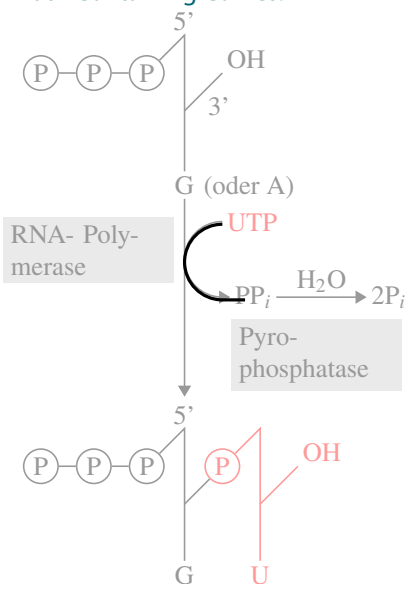



```
\begin{tikzpicture}
  \draw (0mm,54mm)
    -- (5mm,59mm)
    -- (5mm,41mm);
   ...
\end{tikzpicture}
```

- The `\draw` command ist followed by a *path*.
- The path starts with a *coordinate*.
- The path can be continued in straight lines using `--`.

A Path Containing Curves.

2-11

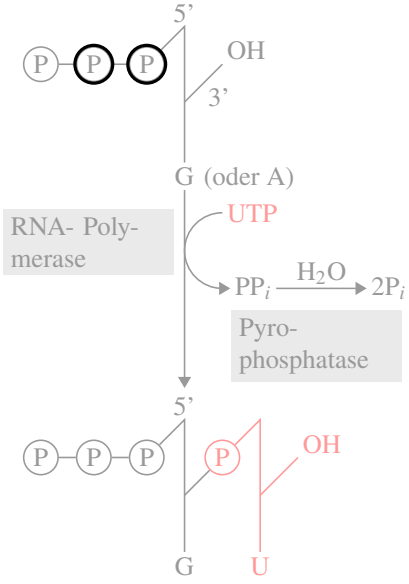



```
\begin{tikzpicture}
  \draw (10mm,34mm)
    arc [start angle=90,
         end angle=270,
         radius=5mm]
     -- ++(3mm,0mm);
   ...
\end{tikzpicture}
```

- An arc can be added to a path using `arc`.
- The parameters of `arc` are
  1. start angle,
  2. end angle and
  3. radius.
- A coordinate prefixed by `++` is relative.

A Path Containg Circles.

2-12

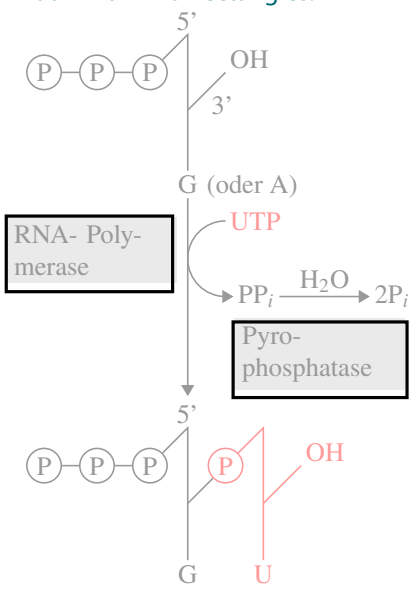


```
\begin{tikzpicture}
  \draw ( 0mm,54mm)
    circle [radius=2.5mm];
  \draw (-7mm,54mm)
    circle [radius=2.5mm];
  ...
\end{tikzpicture}
```

- A circle can be added to a path using `circle`.
- The parameter of a circle are the radius, the center is given by the previous coordinate.

2-13

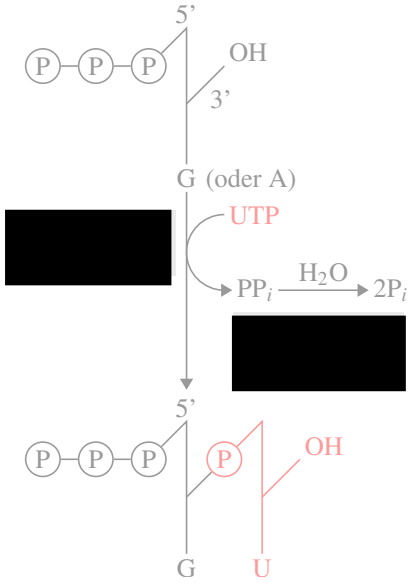
A Path With Two Rectangles.




```
\begin{tikzpicture}
  \draw (-19mm,25mm)
     -- (-19mm,35mm)
     -- (3mm,35mm)
     -- (3mm,25mm)
     -- cycle
           (11mm,21mm)
  rectangle (34mm,11mm);
  ...
\end{tikzpicture}
```

- A path may consist of several parts.
- A part can be closed using `--cycle`.
- A rectangle can be created using `rectangle`.

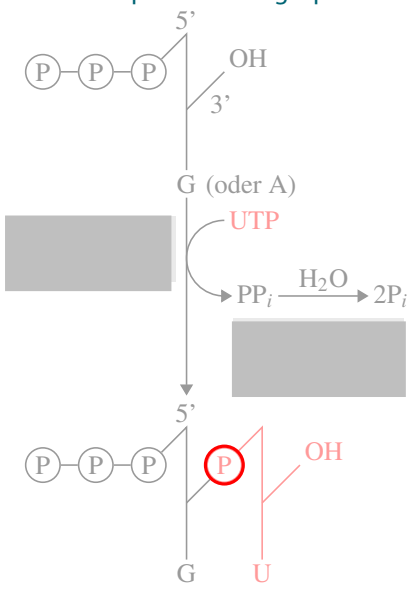
Paths Can be Filled. 2-14




```
\begin{tikzpicture}
  \fill
            (-19mm,25mm)
  rectangle (3mm,35mm)
            (11mm,21mm)
  rectangle (34mm,11mm);
  ...
\end{tikzpicture}
```

- The `\fill` command fills a path.
- It is possible to fill and draw a path.

Colors Are Specified Using Options. 2-15



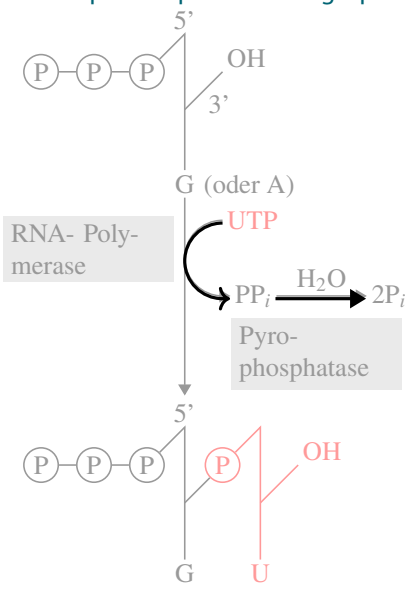

```
\fill[lightgray]
(-19mm,25mm) rectangle ++(22mm,10mm)
(11mm,21mm)  rectangle ++(23mm,-10mm);

\draw[red]
  (10mm,2mm) circle [radius=2.5mm];

...
```

```
\end{tikzpicture}
```

– Colors are specified using options given in square brackets.

2-16

Arrow Tips Are Specified Using Options.

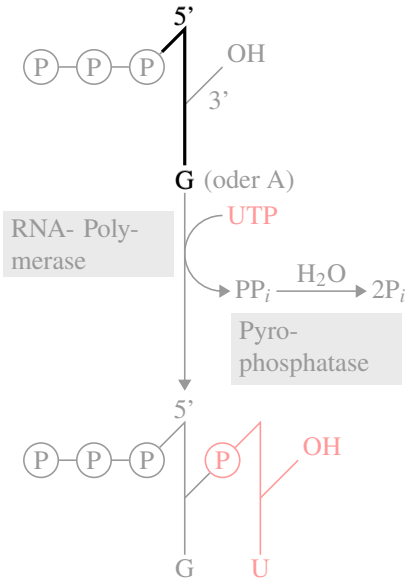



```
\begin{tikzpicture}
  \draw [->]     (10mm,34mm)
   arc [...] -- (11mm,24mm);
  \draw [-Triangle]
     (17mm,24mm)
  -- (27mm,24mm);
  ...
\end{tikzpicture}
```

– Arrow tips are set using an option with a hyphen in the middle.
– Whatever is left of the hyphen specifies the start arrow tip.
– Whatever is right of the hyphen specifies the end arrow tip.
– There are numerous predefined arrow tips.

2-17

Labels Are Added Using Nodes.

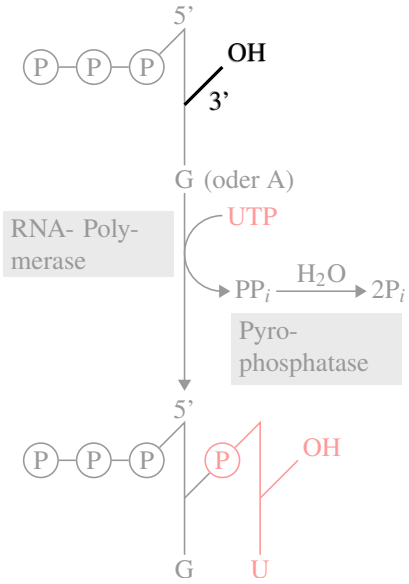


```
\begin{tikzpicture}
  \draw (2mm,56mm)
     -- (5mm,59mm)
        node [above] {5'}
     -- (5mm,41mm)
        node [below] {G};
  ...
\end{tikzpicture}
```

- Nodes are used for adding text.
- The preceding coordinate and options specify the exact placement.
- The node text is given in curly braces.
- Nodes are added after the path has been drawn and filled.

Edge Labels Are Also Added Using Nodes.

2-18

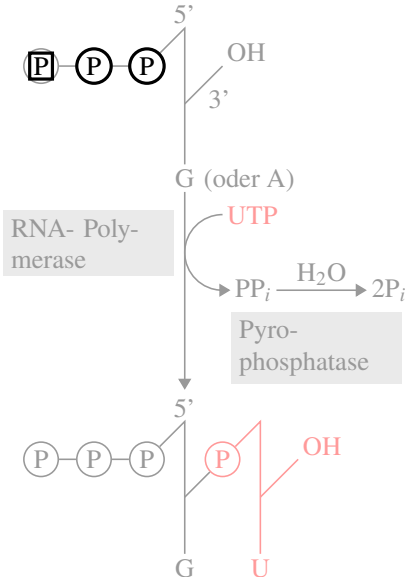



```
\begin{tikzpicture}
  \draw (5mm,49mm) -- (10mm,54mm)
   node [above right] {OH}
   node [midway,
         below right] {3'};
  ...
\end{tikzpicture}
```

- It is possible to add multiple nodes at the same place.
- The `midway` option will place a node at the middle of the previous path segment.

2-19

Nodes Can Have Special Shapes.

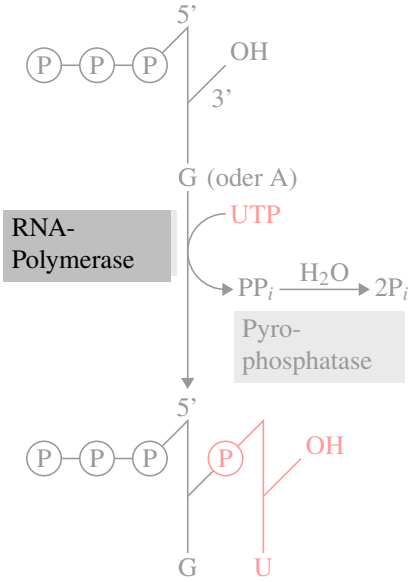



```
\begin{tikzpicture}
  \draw (-14mm,54mm)
    node [draw] {P};

  \draw (-7mm,54mm)
    node [circle,draw]{P};

  \node at (0mm,54mm)
    [circle,draw] {P};
  ...
\end{tikzpicture}
```

- The first path does not contain any lines. Nothing is drawn.
- The `draw` option specifies that the node's shape should be drawn.
- The `circle` specifies a circular shape.
- The `\node` command is just an abbreviation.

2-20

Nodes Can Be Filled.



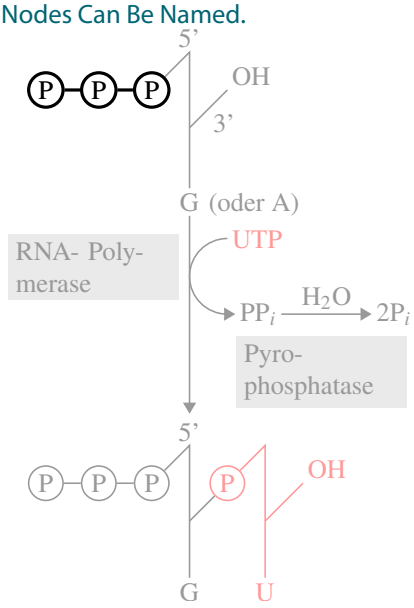

```
\begin{tikzpicture}
  \node at (3mm,35mm)
    [below left,
```

```
    fill=lightgray,
    text width=2cm]
  {RNA-\\Polymerase};
 ...
\end{tikzpicture}
```

- Use `text width` to specify a node's (text) width.
- Use `fill=` to specify a color for filling.

Nodes Can Be Named.

2-21

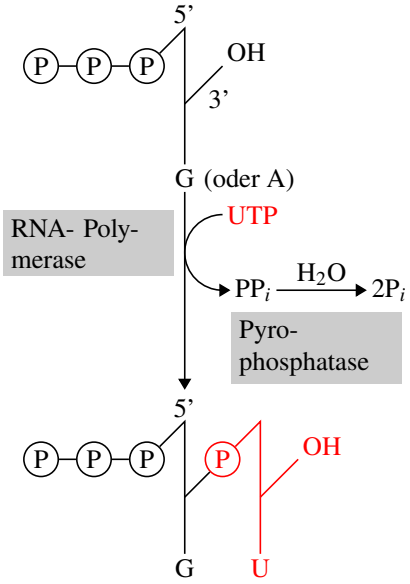



```
\begin{tikzpicture}
  [nodes = {circle, draw}]

  \node at (-14mm,54mm)
    [name=p1] {P};
  \node at (-7mm,54mm)
    [name=p2] {P};
  \node at (0mm,54mm)
    [name=p3] {P};

  \draw (p1) -- (p2) -- (p3);
\end{tikzpicture}
```

- You can assign a name to a node using `name=`.
- Later, a named node can be used "like a coordinate."

2-22

The Complete Picture.



The whole picture can be created using the just-described methods.

2.2 Design-Oriented Overview – Design Principles

2-23

Basic Design Principles Underlying TikZ.

1. Pictures consist of *path*, to which *actions* are applied.
2. Special syntax for *coordinates*.
3. Special syntax for *paths*.
4. Special syntax for *nodes*.
5. Special syntax for *trees*.
6. *Style sheets* configure the way things look.

2.2.1 Paths and Actions

2-24

Design Principle: Paths and Actions
The Concept

Design Principle

All TikZ graphics consist of *paths* to which one or more *actions* are applied.

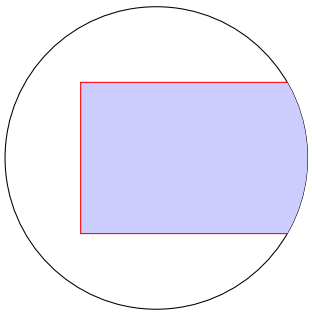
Actions are specified using options:

- `draw` will draw (stroke) a path.
- `fill` will fill a path.
- `shade` will shade the path.
- `pattern` will fill the path using a pattern.
- `clip` will clip the rest of the figure against the path.

The command `\draw` is an abbreviation for `\path[draw]`.

2-25

Design Principle: Paths and Actions
Examples



```
\begin{tikzpicture}
  \path[draw,clip] (0,0) circle [radius=2cm];
  \path[draw=red,fill=blue!20] (-1,-1) rectangle (3,1);
\end{tikzpicture}
```

2.2.2 Special Syntax for Coordinates

Design Principle: Syntax for Coordinates

2-26

The Concept

Design Principle

Coordinates are given in parentheses. Different coordinate systems are possible.

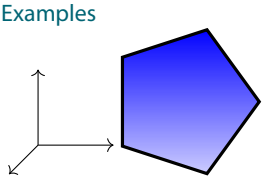
Supported coordinate systems:

- Cartesian
- affine
- polar 2D
- isometric 3D
- barycentric
- user defined

Design Principle: Syntax for Coordinates

2-27

Examples



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

\begin{tikzpicture}
  \draw [top color = blue, bottom color = blue!20,
         draw, very thick]
     (0:1cm) -- (72:1cm) -- (144:1cm)
  -- (216:1cm) -- (288:1cm) -- cycle;
\end{tikzpicture}
```

2.2.3 Special Syntax for Paths

Design Principle: Syntax for Paths

2-28

The Concept

Design Principle

Paths are specified using a sequence of path extension operations.

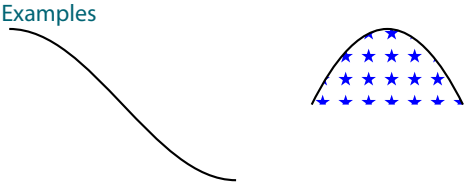
Possible path operations:

- Starting a new path part.
- `--` extends the path in a straight line.
- `arc` extends the path using an arc.
- `..` extends the path using a Bézier curve.
- `parabola` extends the path using a parabola.
- `sin` extends the path using a sine curve.
- `plot` extends the path based on plot data.
- `to` extends the path using a user-defined method.
- ...

2-29

Design Principle: Syntax for Paths

Examples



```
\begin{tikzpicture}[thick]
  \draw (0,1) cos (1.5,0) sin (3,-1);

  \draw [pattern=fivepointed stars,pattern color=blue!80]
        (4,0) parabola[parabola height=1cm] (6,0);
\end{tikzpicture}
```

2.2.4 Special Syntax for Nodes

2-30

Design Principle: Syntax for Nodes

The Concept

Design Principle

Nodes are put at certain places along a path. Nodes have a *shape* and a *text label*.

Possible shapes:

- `rectangle`
- `circle`
- `ellipse`
- `diamond`
- `breakdown diode IEC`
- ...

2-31

Design Principle: Syntax for Nodes

Examples


```
\begin{tikzpicture}
  \node at (0,0)
    [forbidden sign,line width=1ex,draw=red,draw opacity=.8]
      {Smoking};

  \node at (4,0)
    [ellipse,top color=white,bottom color=lightgray]
      {smoke};
\end{tikzpicture}
```

2.2.5 Special Syntax for Graphs

Design Principle: Syntax for Graphs

The Concept

Design Principle

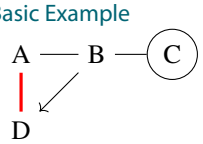
The `graph` operation *switches the syntax locally* to a notation that allows you to specify graphs easily.

Basic ideas:

- All "normal text" gets turned into nodes.
- *Edges* are created by special strings:
  - `--` Undirected edge.
  - `->` A directed edge.
  - `<-` A directed edge, but backward.
  - `<->` Directed edges going in both directions.
  - `-!-` A missing or removed edge.
- Both edges and nodes can have options, which follow in square brackets.

Design Principle: Syntax for Graphs

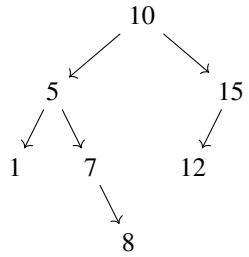
Basic Example




```
\tikz \graph {
  A -- B -- C [draw circle];
  B -> D;
  D --[red, very thick] A
};
```

Design Principle: Syntax for Tree

Complex Example




```
\tikz \graph [binary tree layout] {
  10 -> {
    5 -> {
      1,
      7 -> { , 8 }
    },
    15 -> 12
  }
};
```

2.2.6 Style Sheets

2-35

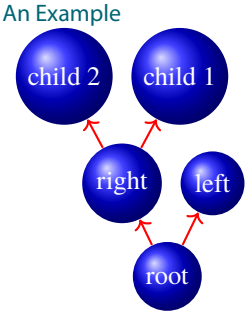
Design Principle: Style Sheets
The Concept

Design Principle
A *style* is a configurable set of options that are automatically or explicitly set in certain situations.

- You define a style named `foo` by saying `foo/.style=some options`.
- Using `foo` anywhere will insert `some options`.
- Styles can use other styles.
- Extensive use of styles makes code more readable and graphics more consistent (similar to HTML and CSS).

2-36

Design Principle: Style Sheets
An Example




```
\tikz [every edge/.style =
         { draw, red, thick },
       every node/.style =
         { circle,
           ball color = blue,
           text = white }]
   \graph [grow = up,
           binary tree layout]
   {
     root -> {
       left,
       right -> {
         child, child
       }
     }
   };
```

2.3 Implementation-Oriented Overview – System Structure

2-37

The Layers Below TikZ.
TikZ is part of the PGF *package* and it just provides a "simple syntax":

1. Top layer: *TikZ Syntax*
   - Easy to use for humans.
   - Succinct.
   - Slow.
2. Middle layer: PGF *base layer*
   - TEX macros for creating figures.
   - Easy to use for other packages.
   - Verbose.
   - Quick.
3. Bottom layer: PGF *system layer*

- Minimalistic set of TEX macros for creating figures.
- Different implementation for each backend driver.
- Extremely difficult to use.
- Extremely fast (as fast as normal TEX).

Let's Trace a Command. 2-38

We trace the following command through the layers:

```
\draw (0,0) -- (30:10pt) -- (60:10pt) -- cycle;
```

It looks like this: 

2.3.1 Top Layer: TikZ

Transformation Done By TikZ. 2-39

The command

```
\draw (0,0) -- (30:10pt) -- (60:10pt) -- cycle;
```

is translated to the following PGF basic layer code by TikZ:

```
\pgfpathmoveto{\pgfpointxy{0}{0}}
\pgfpathlineto{\pgfpointpolar{30}{10pt}}
\pgfpathlineto{\pgfpointpolar{60}{10pt}}
\pgfpathclose
\pgfusepath{draw}
```

2.3.2 Middle Layer: PGF Basic Layer

Transformations Done By the PGF Basic Layer. 2-40

The commands

```
\pgfpathmoveto{\pgfpointxy{0}{0}}
\pgfpathlineto{\pgfpointpolar{30}{10pt}}
\pgfpathlineto{\pgfpointpolar{60}{10pt}}
\pgfpathclose
\pgfusepath{draw}
```

are translated to the following PGF system layer command:

```
\pgfsys@moveto{0pt}{0pt}
\pgfsys@lineto{8.660254pt}{5pt}
\pgfsys@lineto{5pt}{8.660254pt}
\pgfsys@closepath
\pgfsys@stroke
```

2.3.3 Bottom Layer: PGF System Layer

Transformations Done By the PGF System Layer. 2-41

Generation of `special` Commands for `dvips`.

The commands

```
\pgfsys@moveto{0pt}{0pt}
\pgfsys@lineto{8.660254pt}{5pt}
\pgfsys@lineto{5pt}{8.660254pt}
\pgfsys@closepath
\pgfsys@stroke
```

are translated to the following for `dvips`:

```
\special{ps:: 0 0 moveto}
\special{ps:: 8.627899 4.98132 lineto}
\special{ps:: 4.98132 8.627899 lineto}
\special{ps:: closepath}
\special{ps:: stroke}
```

2-42

Transformations Done By the PGF System Layer.

Generation of `special` Commands for `pdftex`.

The commands

```
\pgfsys@moveto{0pt}{0pt}
\pgfsys@lineto{8.660254pt}{5pt}
\pgfsys@lineto{5pt}{8.660254pt}
\pgfsys@closepath
\pgfsys@stroke
```

are translated to the following for `pdftex`:

```
\special{pdf: 0 0 m}
\special{pdf: 8.627899 4.98132 l}
\special{pdf: 4.98132 8.627899 l}
\special{pdf: h}
\special{pdf: S}
```

2-43

Transformations Done By the PGF System Layer.

Generation of `special` Commands for `tex4ht`.

The commands

```
\pgfsys@moveto{0pt}{0pt}
\pgfsys@lineto{8.660254pt}{5pt}
\pgfsys@lineto{5pt}{8.660254pt}
\pgfsys@closepath
\pgfsys@stroke
```

are translated to the following for `tex4ht`:

```
\special{t4ht=<path d="M 0 0
                       L 8.660254 5
                       L 5 8.660254
                       Z"
                     style="stroke">}
```

2.3.4 Gallery of Libraries

2-44

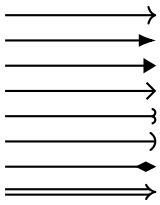
TikZ Comes With Several Libraries

- A *TikZ library* provides *addditional features* or *additional options*.
- You include a library by saying `\usetikzlibrary{some lib}`.
- The list of libraries includes:
  - Additional *arrow tips*.
  - Drawing *automata*, *E/R-diagrams*, *mind maps* and *Petri nets*.
  - Adding *backgrounds* to pictures.
  - Drawing *calendars*.
  - Forming connected *chains* of nodes.
  - *Decorating* paths.
  - Predefined *transparency patterns*.
  - *Fitting* nodes around a set of coordinates.
  - Filling *patterns*.
  - Addditional *shapes*.

Library: arrows.meta

2-45

A Library Defining Additional Arrow Tips

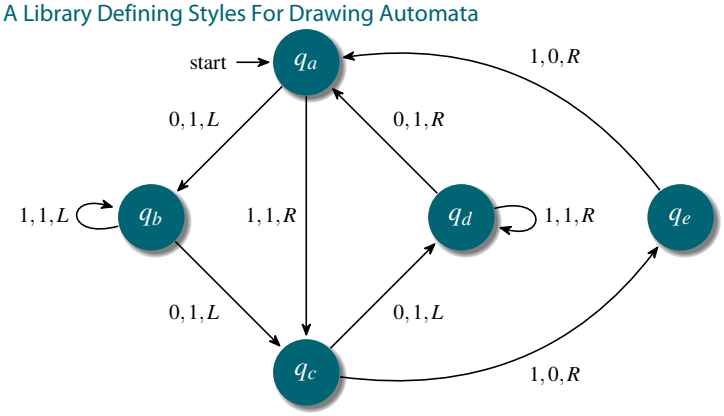


```
\usetikzlibrary{arrows}
...
\draw[-To]               (0,7) -- (2,7);
\draw[-LaTeX]            (0,6) -- (2,6);
\draw[-Triangle]         (0,5) -- (2,5);
\draw[-Straight Barb]    (0,4) -- (2,4);
\draw[-Hooks]            (0,3) -- (2,3);
\draw[-)]                (0,2) -- (2,2);
\draw[-Diamond]          (0,1) -- (2,1);
\draw[-Implies,double]   (0,0) -- (2,0);
```

Library: automata

2-46

A Library Defining Styles For Drawing Automata



Library: automata

2-47

A Library Defining Styles For Drawing Automata

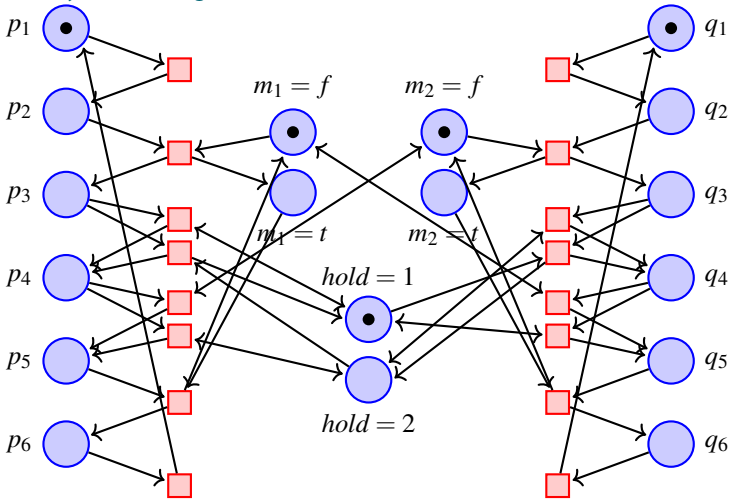
```
\usetikzlibrary{automata}
\begin{tikzpicture}
   [->,auto=right, node distance=2cm,
    >={Stealth[round,sep]}, semithick,
    every state/.style={draw=none, fill=structure.fg,
                        text=white, circular drop shadow},
    every edge/.style={font=\footnotesize, draw}]

  \node[initial,state] (q_a)                       {$q_a$};
  \node[state]         (q_b) [below left=of q_a]   {$q_b$};
  \node[state]         (q_d) [below right=of q_a]  {$q_d$};
  \node[state]         (q_c) [below right=of q_b]  {$q_c$};
  \node[state]         (q_e) [right=of q_d]        {$q_e$};

  \draw (q_a) edge              node {$0,1,L$} (q_b)
              edge              node {$1,1,R$} (q_c)
        (q_b) edge [loop left]  node {$1,1,L$} (q_b)
              edge              node {$0,1,L$} (q_c)
        (q_c) edge              node {$0,1,L$} (q_d)
```

```
                edge [bend right] node {$1,0,R$} (q_e)
        (q_d) edge [loop right] node {$1,1,R$} (q_d)
              edge              node {$0,1,R$} (q_a)
        (q_e) edge [bend right] node {$1,0,R$} (q_a);
\end{tikzpicture}
```

2-48

Library: petri

A Library For Drawing Petri Nets



2-49

Library: petri

A Library For Drawing Petri Nets

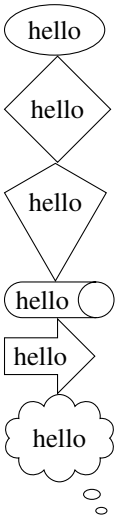
```
\usetikzlibrary{petri}
...
\node[place,label=left:$p_1$,tokens=1] (p1) at (0,1) {};
\node[place,label=left:$p_2$,tokens=0] (p2) at (0,2) {};
...
\node[transition] at (1.5,1.5) {}  edge [pre]  (p1)
                                   edge [post] (p2);
\node[transition] at (1.5,2.5) {}  edge [pre]  (p2)
                                   edge [pre]  (m1f)
                                   edge [post] (p3)
                                   edge [post] (m1t);
\node[transition] at (1.5,3.3) {}  edge [pre] (p3)
                                   edge [post] (p4)
                                   edge [pre and post] (h1);
```

Libraries: shapes

2-50

A Set of Libraries Defining New Shapes

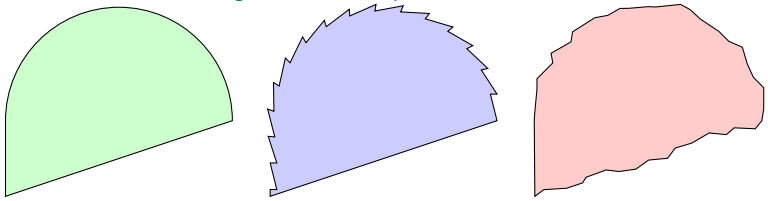
```
\node[draw,ellipse] {hello};
\node[draw,diamond] {hello};
\node[draw,kite] {hello};
\node[draw,cylinder] {hello};
\node[draw,single arrow] {hello};
\node[draw,cloud callout] {hello};
```



Libraries: decorations

2-51

Libraries For "Decorating" Paths In Complex Manners.

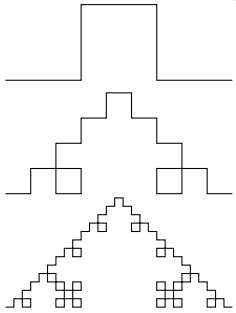


```
\begin{tikzpicture}
  \draw [fill=green!20]
    (0,0) -- (3,1) arc (0:180:1.5) -- cycle;

  \draw [fill=blue!20,xshift=3.5cm,
         decoration=saw]
    (0,0) -- (3,1) decorate { arc (0:180:1.5) -- cycle};

  \draw [fill=red!20,xshift=7cm,
         decoration={random steps,segment length=2mm}]
    decorate { (0,0) -- (3,1) arc (0:180:1.5)} -- cycle;
\end{tikzpicture}
```

Libraries: decorations

2-52

Libraries For "Decorating" Paths In Complex Manners.



```
\begin{tikzpicture}[decoration=Koch curve type 1]
  \draw decorate{ (0,0) -- (3,0) };
  \draw decorate{ decorate{ (0,-1.5) -- (3,-1.5) }};
  \draw decorate{ decorate{ decorate{ (0,-3) -- (3,-3) }}};
\end{tikzpicture}
```

Chapter Summary

2-53

1. TikZ provides a set of *TEX macros* for creating figures directly inside TEX.
2. TikZ works with all *standard backend drivers and formats*.
3. TikZ has a *powerful, consistent syntax*.
4. TikZ is especially suited for *small* or *highly structured figures*.

3-1

Chapter 3
Creating Graphics From Scratch
Case Studies

Chapter Objectives

1. You can create complex figures using Ti*k*Z.
2. You can identify problematic details in existing pictures.
3. You know strategies for drawing these details correctly.

3-2

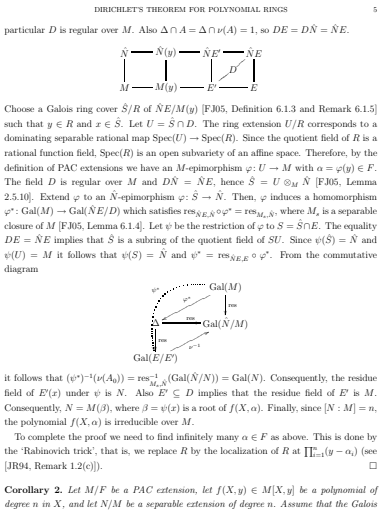
Chapter Contents

3.1 Figure 1: Commutative Diagram

3.1.1 The Figure and a Critique

3-4

A Page From a GTEM Publication with a Figure.



DIRICHLET'S THEOREM FOR POLYNOMIAL RINGS 5

particular D is regular over M. Also $\Delta \cap A = \Delta \cap \nu(A) = 1$, so $DE = D\hat{N} = \hat{N}E$.

$$\begin{array}{ccccc} \hat{N} & - & \hat{N}(y) & - & \hat{N}E' & - & \hat{N}E \\ | & & | & & | & D & | \\ M & - & M(y) & - & E' & - & E \end{array}$$

Choose a Galois ring cover $\hat{S}/R$ of $\hat{N}E/M(y)$ [FJ05, Definition 6.1.3 and Remark 6.1.5] such that $y \in R$ and $x \in \hat{S}$. Let $U = \hat{S} \cap D$. The ring extension U/R corresponds to a dominating separable rational map $\mathrm{Spec}(U) \to \mathrm{Spec}(R)$. Since the quotient field of R is a rational function field, $\mathrm{Spec}(R)$ is an open subvariety of an affine space. Therefore, by the definition of PAC extensions we have an M-epimorphism $\varphi\colon U \to M$ with $\alpha = \varphi(y) \in F$. The field D is regular over M and $D\hat{N} = \hat{N}E$, hence $\hat{S} = U \otimes_M \hat{N}$ [FJ05, Lemma 2.5.10]. Extend φ to an $\hat{N}$-epimorphism $\varphi\colon \hat{S} \to \hat{N}$. Then, φ induces a homomorphism $\varphi^*\colon \mathrm{Gal}(M) \to \mathrm{Gal}(\hat{N}E/D)$ which satisfies $\mathrm{res}_{\hat{N}E,\hat{N}} \circ \varphi^* = \mathrm{res}_{M_s,\hat{N}}$, where M_s is a separable closure of M [FJ05, Lemma 6.1.4]. Let ψ be the restriction of φ to $S = \hat{S} \cap E$. The equality $DE = \hat{N}E$ implies that $\hat{S}$ is a subring of the quotient field of SU. Since $\psi(\hat{S}) = \hat{N}$ and $\psi(U) = M$ it follows that $\psi(S) = \hat{N}$ and $\psi^* = \mathrm{res}_{\hat{N}E,E} \circ \varphi^*$. From the commutative diagram

it follows that $(\psi^*)^{-1}(\nu(A_0)) = \mathrm{res}^{-1}_{M_s,\hat{N}}(\mathrm{Gal}(\hat{N}/N)) = \mathrm{Gal}(N)$. Consequently, the residue field of $E'(x)$ under ψ is N. Also $E' \subseteq D$ implies that the residue field of E' is M. Consequently, $N = M(\beta)$, where $\beta = \psi(x)$ is a root of $f(X, \alpha)$. Finally, since $[N : M] = n$, the polynomial $f(X, \alpha)$ is irreducible over M.

To complete the proof we need to find infinitely many $\alpha \in F$ as above. This is done by the 'Rabinovich trick', that is, we replace R by the localization of R at $\prod_{i=1}^{n}(y - \alpha_i)$ (see [JR94, Remark 1.2(c)]). □

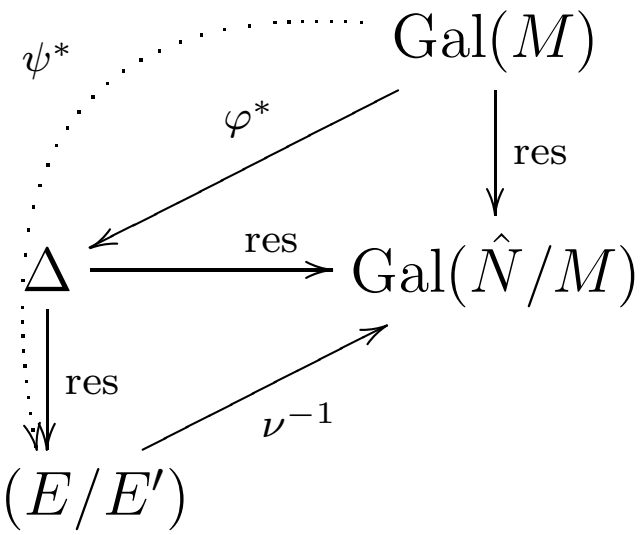
Corollary 2. *Let M/F be a PAC extension, let $f(X, y) \in M[X, y]$ be a polynomial of degree n in X, and let N/M be a separable extension of degree n. Assume that the Galois*

References

[1] Bary-Soroker Lior Dirichlet's Theorem For Polynomial Rings arXiv:math/0612801v2

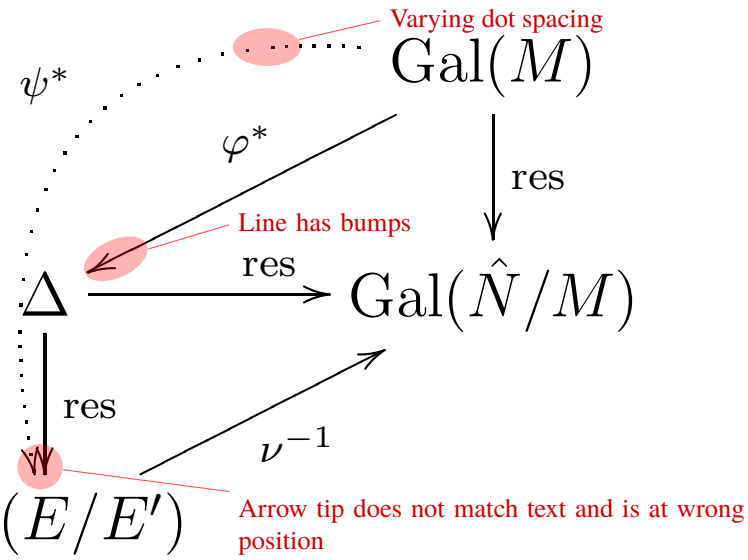
3-5

Closeup of the Figure



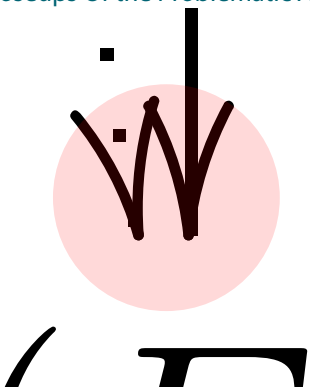
Critique

3-6

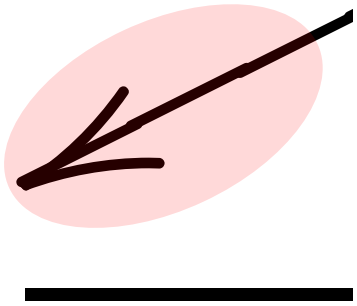


Closeups Of the Problematic Areas.

3-7



Arrow tip does not match text and is at wrong position



Line has bumps

3.1.2 Step 1: The Nodes

3-8

Step 1: Creating the Nodes.

Basic Idea

To (re)create the figure in Ti*k*Z, we start with the *nodes*, which are created using the `node` command.

Syntax of the Node Creation Command

- Start with `\node`.
- Then comes a sequences of *options*.
- Options are given in square brackets, with two exceptions:
  We can say `at (coordinate)` to specify a special place, where the node should go.
  We can say `(name)` to assign a name to a node.
- The node ends with some text in curly braces.

3-9

Step 1: Creating the Nodes.

A Simple Placement

$\mathrm{Gal}(M)$

Δ $\mathrm{Gal}(\hat{N}/M)$

(E/E')

```
\begin{tikzpicture}
  \node (EE)     at (0,0)    {$(E/E')$};
  \node (Delta)  at (0,1.5)  {$\Delta$};
  \node (GalNM)  at (3,1.5)  {$\mathrm{Gal}(\hat N/M)$};
  \node (GalM)   at (3,3)    {$\mathrm{Gal}(M)$};
\end{tikzpicture}
```

3-10

Step 1: Aligning the Nodes

Basic Idea.

The Problem

Providing "hard-wired" coordinates like `(3,1.5)` is *problematic*:

- When you read the code, it is hard to tell, where something will go.
- When you change something later, you may need to change many such coordinates.
- It is hard to make sure that all spacings and alignments are correct.

Possible Solutions

- You can use options like `right=of Delta` to place a node relative to some other node.
- You can use a *TikZ-matrix*. It works like a LaTeX matrix, only inside a picture.

3-11

Step 1: Aligning the Nodes.

Alignment Using a Matrix.

$\mathrm{Gal}(M)$

Δ $\mathrm{Gal}(\hat{N}/M)$

(E/E')

```
\matrix[column sep=1cm,row sep=1cm]
{
                          & \node (GalM)  {$\Gal(M)$};       \\
  \node (Delta) {$\Delta$};& \node (GalNM) {$\Gal(\hat N/M)$};\\
  \node (EE)    {$(E/E')$};&                                 \\
};
```

Step 1: Aligning the Nodes.

3-12

Simplified Version. . .

$$\mathrm{Gal}(M)$$

$$\Delta \qquad\qquad \mathrm{Gal}(\hat{N}/M)$$

$$(E/E')$$

```
\matrix [column sep=1cm,row sep=1cm,matrix of math nodes]
    (fig)
{
          & \Gal(M)          \\
  \Delta  & \Gal(\hat N/M)   \\
  (E/E')  &                          \\
};
% Reference Gal(M) as (fig-1-2)
```

Step 1: Aligning the Nodes.

3-13

. . . With Alternate Naming of Nodes.

$$\mathrm{Gal}(M)$$

$$\Delta \qquad\qquad \mathrm{Gal}(\hat{N}/M)$$

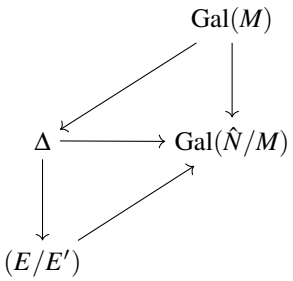
$$(E/E')$$

```
\matrix [column sep=1cm,row sep=1cm,matrix of math nodes]
{
                     & |(M)|   \Gal(M)          \\
  |(Delta)| \Delta   & |(NM)| \Gal(\hat N/M)   \\
  |(EE)|     (E/E')  &                                  \\
};
% Reference Gal(M) as (M)
```

3.1.3 Step 2: The Edges

3-14

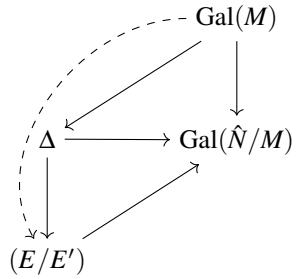
Step 2: Connecting the Nodes.
Simple Straight Line.




```
\matrix [column sep=1cm,row sep=1cm,matrix of math nodes]
{
                    &  |(M)|   \Gal(M)          \\
  |(Delta)| \Delta   &  |(NM)|  \Gal(\hat N/M)  \\
  |(EE)|    (E/E')   &                          \\
};
\draw (M)      edge [->] (Delta)
               edge [->] (NM)
      (Delta)  edge [->] (NM)
               edge [->] (EE)
      (EE)     edge [->] (NM);
```

3-15

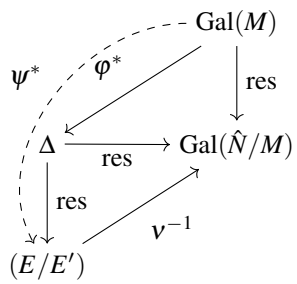
Step 2: Connecting the Nodes.
The Curved, Dashed Line.




```
\draw (M)      edge [->] (Delta)
               edge [->] (NM)
               edge [->,dashed,out=180,in=120] (EE)
      (Delta)  edge [->] (NM)
               edge [->] (EE)
      (EE)     edge [->] (NM);
```

3-16

Step 2: Connecting the Nodes.
Adding the Labels

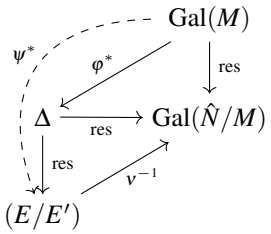


```
\draw [auto=right]
  (M)      edge [->] node {$\varphi^*$}      (Delta)
           edge [->] node [swap] {res}       (NM)
           edge [->,dashed,out=180,in=120]
                     node {$\psi^*$}         (EE)
  (Delta)  edge [->] node {res}              (NM)
           edge [->] node [swap] {res}       (EE)
  (EE)     edge [->] node {$\nu^{-1}$}       (NM);
```

3.1.4 Step 3: Finishing Touches

Step 3: Finishing Touches

3-17



- Adjust “looseness” of the curve and dash phase.
- Reduce distance of φ^*, ψ^* and ν^{-1} to the line.
- Make edge labels smaller (as in $A \xrightarrow{X} B$)

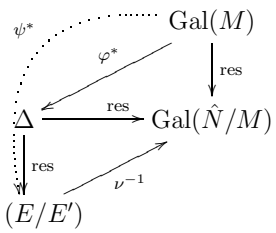
The Complete Code.

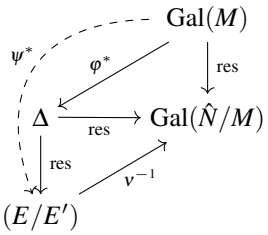
3-18

```
\begin{tikzpicture}
  \matrix [column sep=7mm,row sep=7mmm,matrix of math nodes]
  {
                      & |(M)|   \Gal(M)          \\
    |(Delta)| \Delta  & |(NM)|  \Gal(\hat N/M)   \\
    |(EE)|    (E/E') &                          \\
  };
  \draw [auto=right,nodes={font=\scriptsize}]
    (M)      edge [->] node [inner sep=0pt] {$\varphi^*$} (Delta)
             edge [->] node [swap]          {res}         (NM)
             edge [->,out=180,in=110,looseness=1.4,
                   dashed,dash phase=3pt]
                   node [inner sep=0pt] {$\psi^*$}    (EE)
    (Delta)  edge [->] node               {res}         (NM)
             edge [->] node [swap]        {res}         (EE)
    (EE)     edge [->] node [inner sep=0pt] {$\nu^{-1}$} (NM);
\end{tikzpicture}
```

Comparison of Original and Reworked Figure.

3-19



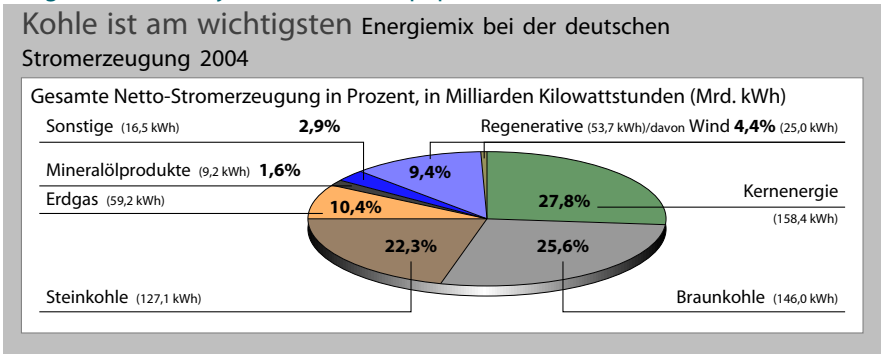


3.2 Figure 2: A Pie Chart

3.2.1 The Figure and a Critique

3-20

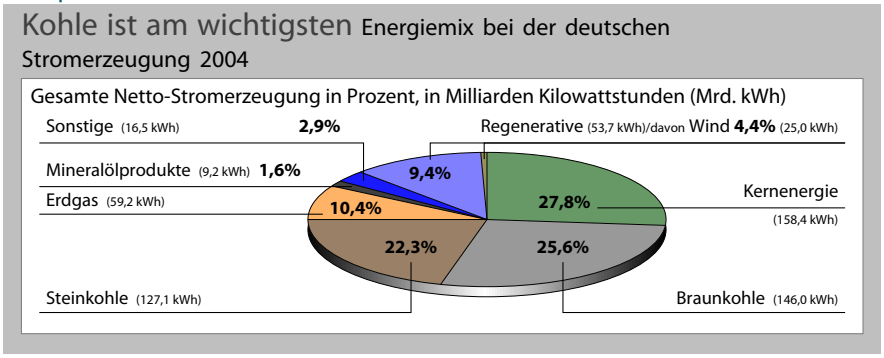
A Figure From a Major German Newspaper.



This figure is a redrawing of a figure from “Die Zeit,” June 4th, 2005.

3-21

Critique.

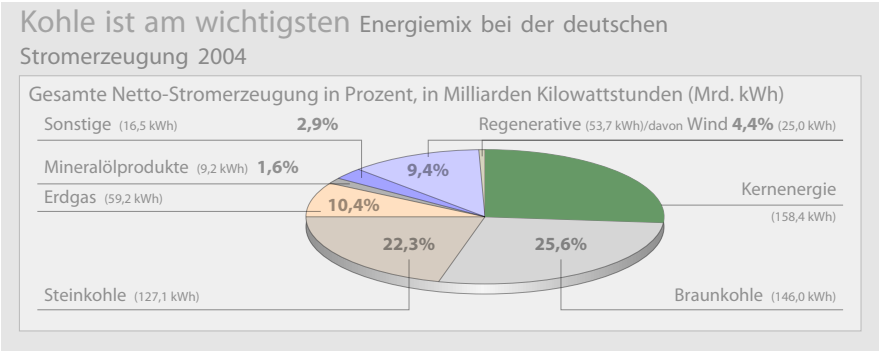


- Coloring is random and misleading.
- Pie slice sizes do not reflect percentages.
- Main message is lost since coal is split across page.

3.2.2 Detail 1: Elliptical Arcs

3-22

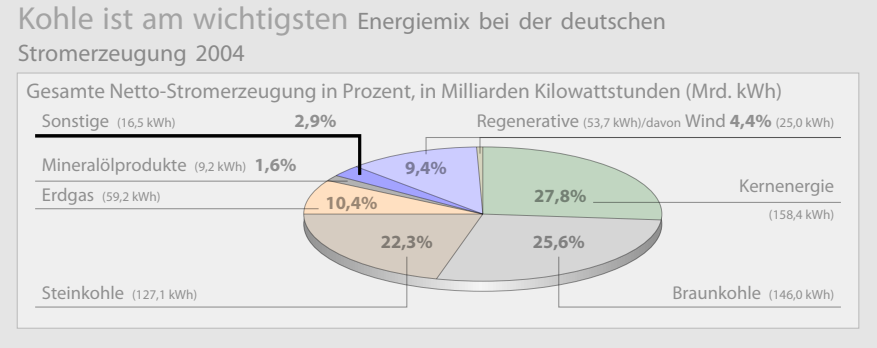
Detail 1: Pie Slices are Elliptic Arcs.




```
\fill[green!20!gray]
     (0,0)
  -- (90:1.2cm)
  arc[start angle=90, end angle=-5,
      x radius=3.2cm, y radius=1.2cm]
  -- cycle;
```

3.2.3 Detail 2: Perpendicular Lines

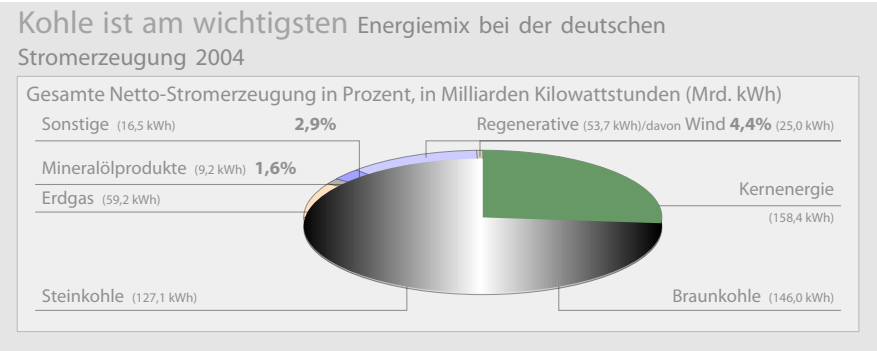
Detail 2: A Horizontal/Vertical Junction. 3-23




```
\draw[very thick] (-22mm,7mm) |- (-80mm,14mm);
```

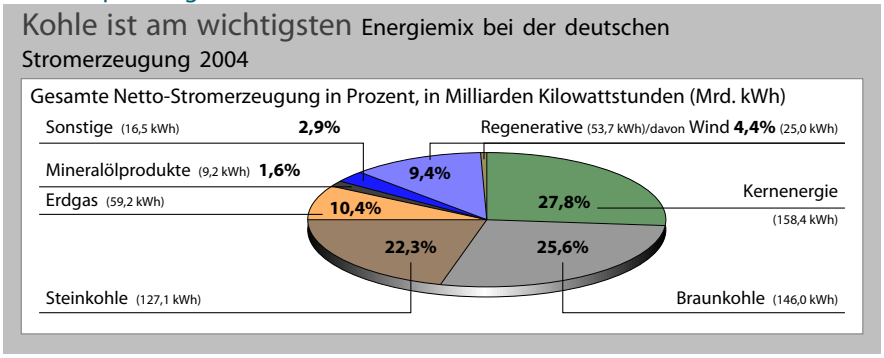
3.2.4 Detail 3: Shadings

Detail 3: The Shading in the Pie Chart. 3-24




```
\shade [left color=black,right color=black,middle
   color=white]
  (0mm,-1.5mm) ellipse [x radius=3.2cm, y radius=1.2cm];

\fill[green!20!gray]
     (0,0)
  -- (90:1.2cm)
  arc[start angle=90, end angle=-5,
      x radius=3.2cm, y radius=1.2cm]
  -- cycle;
```

The Complete Figure. 3-25



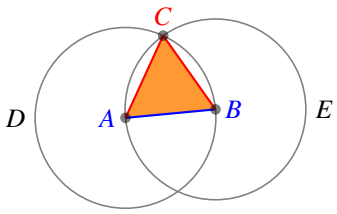
The complete figure can be constructed in this way.

3.3 Figure 3: A Construction From Euclid's Elements

3.3.1 The Figure

3-26

A Geometrical Construction



References

[1] Euclid of Alexandria Proof of Proposition I Elements, Book I

3.3.2 Step 1: The Line *AB*

3-27

Step 1: The Line *AB*

A Simple Line

```
\begin{tikzpicture}
  \coordinate (A) at (0,0);
  \coordinate (B) at (1.25,0.25);

  \draw[blue] (A) -- (B);
\end{tikzpicture}
```

– The `\coordinate` command is a shorthand for the `\node` command with empty text.

3-28

Step 1: The Line *AB*

Adding Labels


```
\begin{tikzpicture}
  \coordinate [label=left:\textcolor{blue}{$A$}]
    (A) at (0,0);

  \coordinate [label=right:\textcolor{blue}{$B$}]
    (B) at (1.25,0.25);

  \draw[blue] (A) -- (B);
\end{tikzpicture}
```

– The `label` option makes it easy to add some text *around another node*.
– Alternatively, one could explicitly create a node later on.

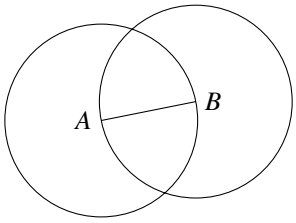
Step 1: The Line AB 3-29
Perturbed Positions


```
\usetikzlibrary{calc}
\begin{tikzpicture}
  \coordinate [label=left:\textcolor{blue}{$A$}]
    (A) at ($ (0,0) + .1*(rand,rand) $);

  \coordinate [label=right:\textcolor{blue}{$B$}]
    (B) at ($ (1.25,0.25) + .1*(rand,rand) $);

  \draw[blue] (A) -- (B);
\end{tikzpicture}
```

– Between `($` and `$)` you can do some *basic linear algebra on coordinates*.

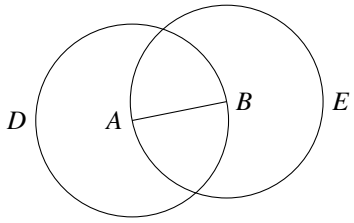
3.3.3 Step 2: The Circles

Step 2: The Circles 3-30
Using the Let Operation




```
...
\draw (A) -- (B);

\draw let
        \p1         = ($ (B) - (A) $)
      in
        (A) circle [radius={sqrt(\x1*\x1+\y1*\y1)}]
        (B) circle [radius={sqrt(\x1*\x1+\y1*\y1)}];
```

Step 2: The Circles 3-31
Using the Through Library

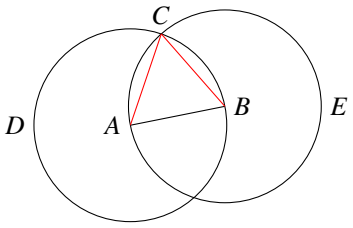



```
\usetikzlibrary{through}
...
\draw (A) -- (B);

\node at (A) [draw,circle through=(B),label=left:$D$]  {};
\node at (B) [draw,circle through=(A),label=right:$E$] {};
```

3.3.4 Step 3: The Intersection of the Circles

3-32

Step 3: The Intersection of the Circles

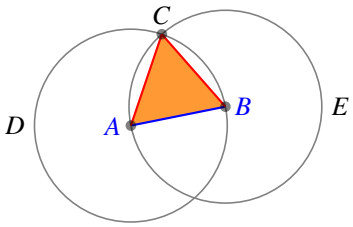



```
\usetikzlibrary{intersections}
...
\draw (A) -- (B);
\node at (A) [name path=D,draw,circle
   through=(B),label=...] {};
\node at (B) [name path=E,draw,circle
   through=(A),label=...] {};

\node [name intersections={of=D and E, by=C}]
   at (C) [above] {$C$};

\draw [red] (A) -- (C)    (B) -- (C);
```

3.3.5 Step 4: Finishing Touches

3-33

Step 4: Finishing Touches



- Add *transparent circles* at the points A, B, and C.
- Fill triangle, but on the *background layer*.

3-34

The Complete Code

```
\begin{tikzpicture}
    [thick, help lines/.style={semithick,draw=black!50}]
  \coordinate [label=left:\textcolor{blue}{$A$}]
    (A) at ($ (0,0) + .1*(rand,rand) $);
  \coordinate [label=right:\textcolor{blue}{$B$}]
    (B) at ($ (1.25,0.25) + .1*(rand,rand) $);
  \draw [blue] (A) -- (B);

  \node at (A) [circle through=(B),name path=D,
                help lines,draw,label=left:$D$] {};
  \node at (B) [circle through=(A),name path=E,
                help lines,draw,label=right:$E$] {};

  \node [name intersections={of=D and E, by=C}]
     at (C) [above] {$C$};
  \draw [red] (A) -- (C)    (B) -- (C);

  \foreach \point in {A,B,C}
    \fill [black,opacity=.5] (\point) circle (2pt);
```

```
  \begin{pgfonlayer}{background}
    \fill[orange!80] (A) -- (C) -- (B) -- cycle;
  \end{pgfonlayer}
\end{tikzpicture}
```

Chapter Summary

- Ti*k*Z offers different ways of achieving the same results.
- It pays to pay attention to details in figures.

3-35