# Wissenschaftl. Textverarbeitung mit LaTeX
# WS 2015/16 - 6. Vorlesung

Alexander Richter

Institut für Mathematische Optimierung

7. Dezember 2015

# Rückblick

In der vorletzten Woche wurde u.a. behandelt:

- Tabellen III
- Eigene Zähler
- Eigene Umgebungen

# Lösung der Hausaufgabe

~~Lösung der Hausaufgabe.~~
(Auf nächste Woche verschoben)

# Tagesprogramm

1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

# Tagesprogramm

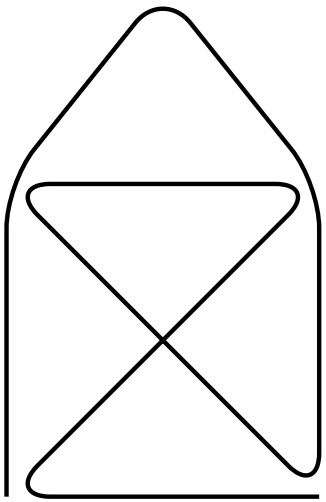
1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

# TikZ: Was ist das?

- TikZ ist kein Zeichenprogramm (kein WYSIWG)

- Befehle liefern eine Zeichenvorschrift zur Erstellung von Graphiken
- LaTeX-Code kann mit (fast) allen Funktionalitäten in sog. TikZ-Nodes verwendet werden
- LaTeX: “everything is a box”, TikZ: “everything is a path”



```
\tikz \draw[thick,rounded corners=8pt]
(0,0) -- (0,2) -- (1,3.25) -- (2,2) --
(2,0) -- (0,2) -- (2,2) -- (0,0) -- (2,0);
```

# TikZ: Was ist das?

- TikZ ist ein Paket, das auf `pgf` aufbaut
- `pgf` ist ein Paket
  - zur Erzeugung grafischer Darstellungen
  - unabhängig vom Ausgabeformat

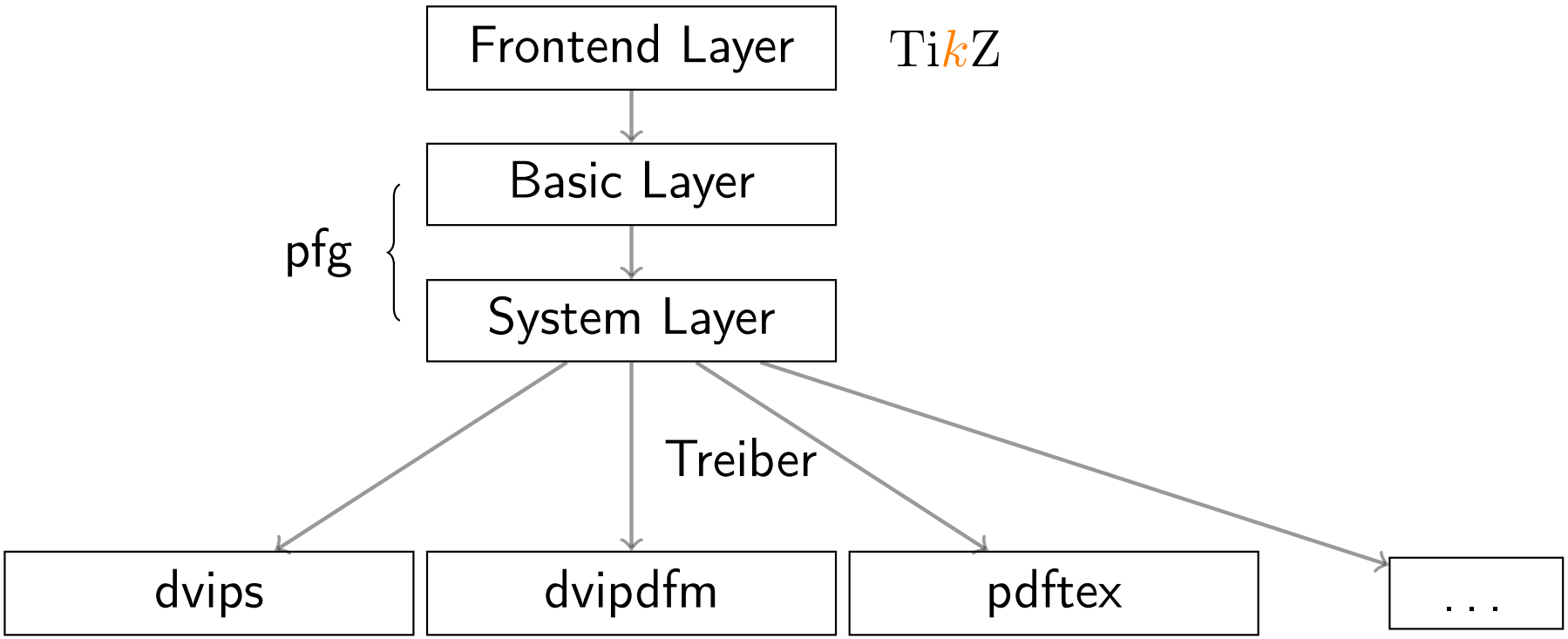
  **Achtung!** Es wird eine aktuelle Version benötigt (mindestens $\geq 2.0$ )

  Meine Version: 3.0.1a

  ```
  Meine Version:  \pgfversion
  ```

**Warnung!** Spätestens ab jetzt: nur pdflatex

# TikZ: Einführung



# Beispiele

## Frontend Layer

```
\path[draw](10pt,10pt)--(0,0);
```

## Basic Layer

```
\pgfpathmoveto{\pgfpoint{10pt}{10pt}}
\pgfpathlineto{\pgfpointorigin}
\pgfusepath{stroke}
```

## System Layer

```
\pgfsys@moveto{10pt}{10pt}
\pgfsys@lineto{0pt}{0pt}
\pgfsys@stroke
```

# Benutzung

Laden des Pakets:

- `\usepackage{tikz}`
- zusätzliche TikZ-Bibliotheken:
  `\usetikzlibrary{ <lib1>, <lib2>, ...}`

Beispiele für Bibliotheken:

- backgrounds
- scopes,
- shapes.geometric
- shadows
- decorations.shapes
- decorations.pathmorphing
- graphs
- calc
- intersections
- ...

# Benutzung

Laden des Pakets:

- `\usepackage{tikz}`
- zusätzliche TikZ-Bibliotheken:
  `\usetikzlibrary{ <lib1>, <lib2>, ...}`

Beispiele für Bibliotheken:

- backgrounds
- scopes,
- shapes.geometric
- shadows
- decorations.shapes
- decorations.pathmorphing
- graphs
- calc
- intersections
- ...

# Benutzung

- zwei Möglichkeiten, ein TikZ-Bild zu erzeugen:

```
\begin{tikzpicture}
   <Pfad 1>;
   <Pfad 2>;
    ...
   <Pfad 2>;
\end{tikzpicture}
```

```
\tikz{
   <Pfad 1>;
   <Pfad 2>;
    ...
   <Pfad 2>;
}
```

# Tagesprogramm

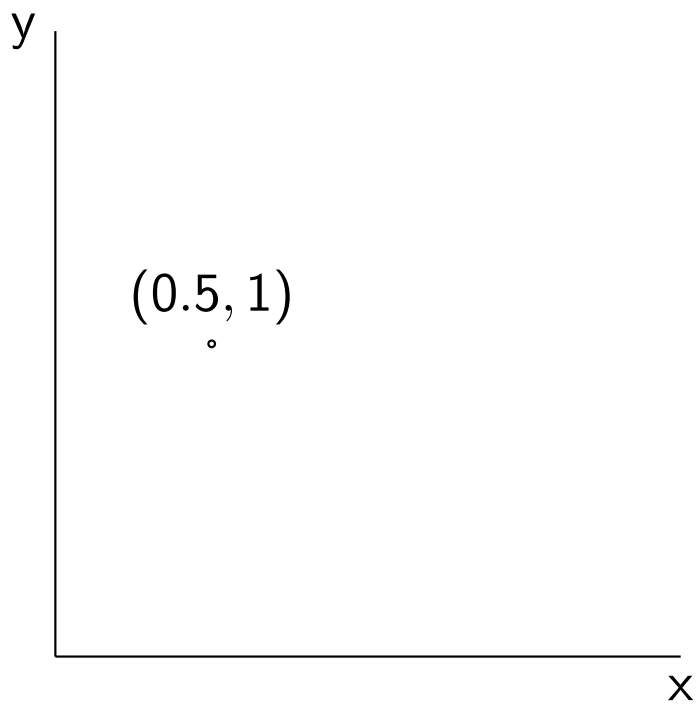
1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

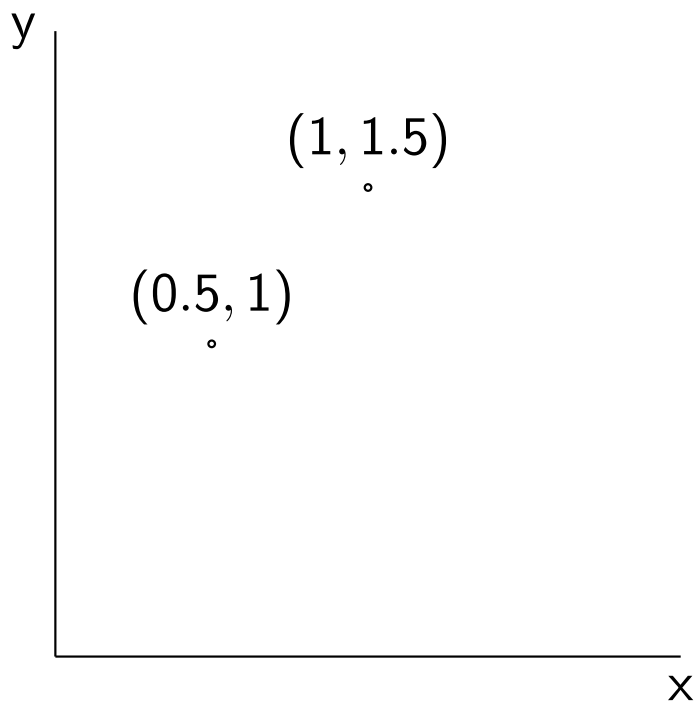
# Konzept: Koordinaten

- Euklidisches Koordinatensystem 2D
- Euklidisches Koordinatensystem 3D
- Polarkoordinaten (2D und 3D)
- abstrakte Koordinatensysteme
- ...

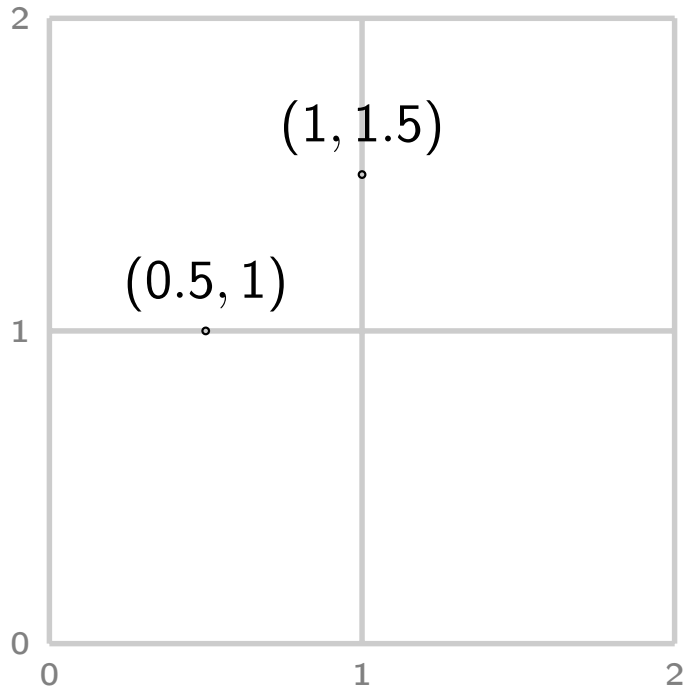
# Euklidische Koordinaten



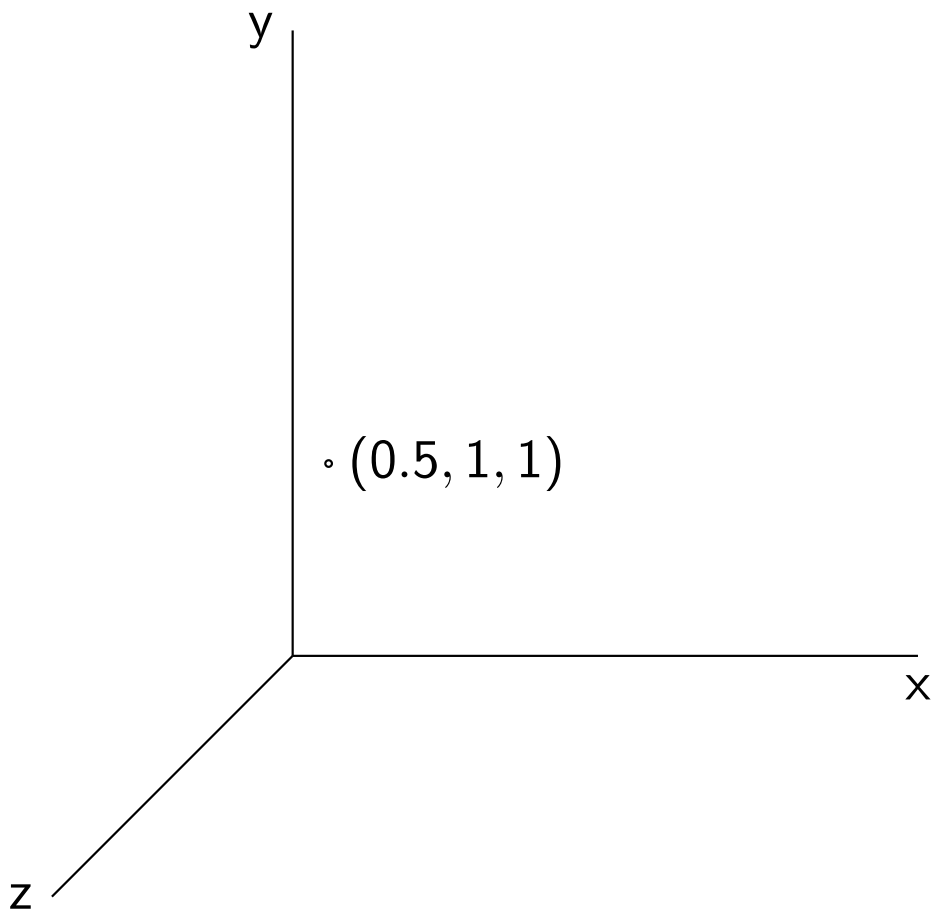
# Euklidische Koordinaten



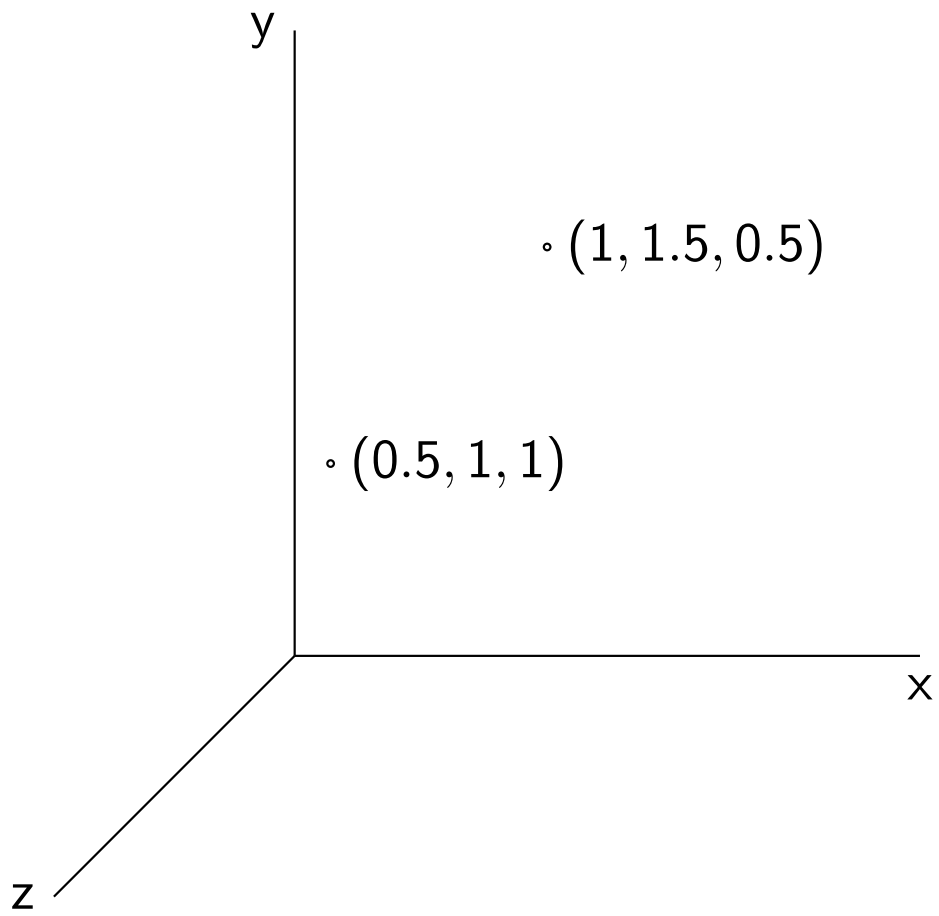
# Euklidische Koordinaten



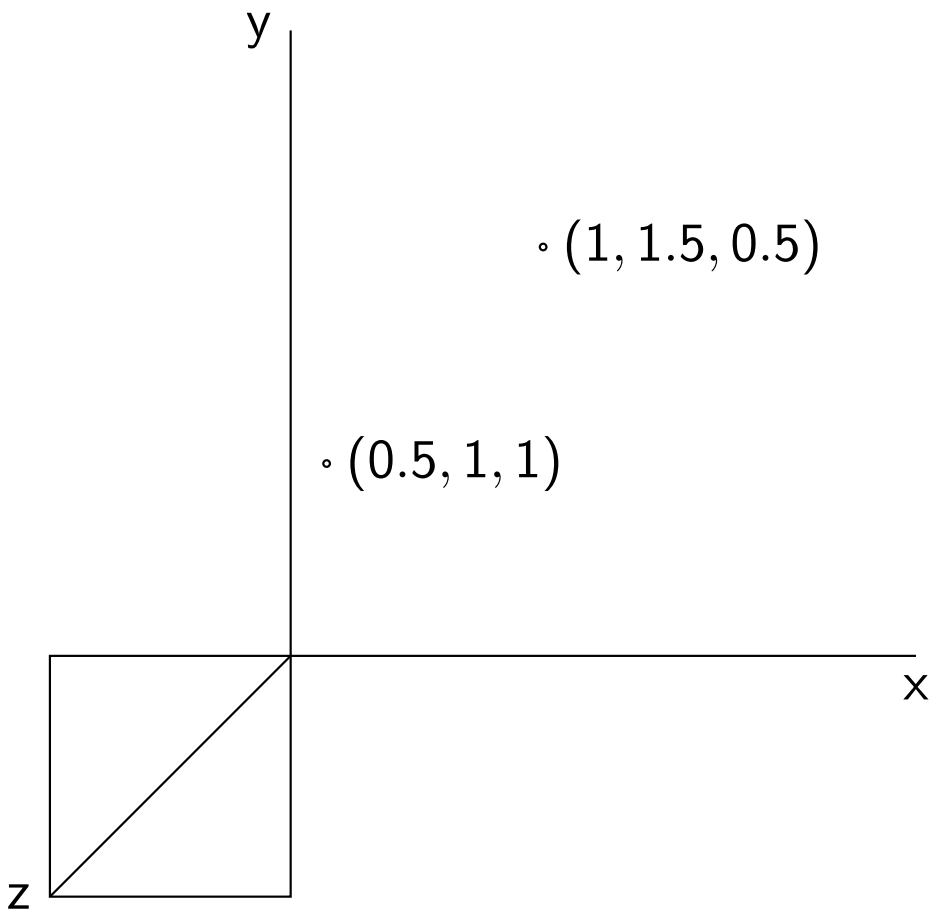
# Euklidische Koordinaten



# Euklidische Koordinaten



# Euklidische Koordinaten



# Tagesprogramm

1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

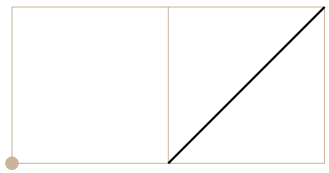
# Konzept: Pfade, Pfadaktionen, Pfadoperationen

```
\path[<Pfadaktionen>]<Pfadoperationen> ;  % Semi-
% kolon NICHT vergessen!!
```

- *Bild*: eine Folge von Pfaden
- *Pfad*:
  - Darstellung wird durch *Pfadaktionen* spezifiziert
  - Pfad wird aus *Pfadoperationen* konstruiert
  - Pfadoperationen werden mit *Punkten* in einem Koordinatensystem spezifiziert

# Pfade

**Beispiel:**



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

- Pfadaktion: `draw`
- Pfadoperation 1: `(1cm,0cm)` “move-to”-Operation
- Pfadoperation 2: `--(1cm,0cm)` “line-to”-Operation

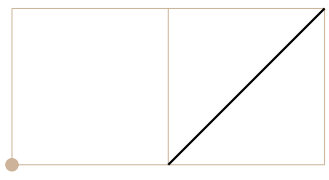
# fehlende Pfadoperationen

Ohne Pfadaktion wird nur der Platz bereitgestellt

```
\path (0cm,0cm) (1cm,0cm) -- (2cm, 1cm);
```

# fehlende Längeneinheiten

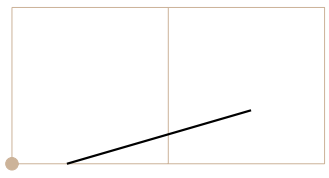
cm-Angaben sind der Default Wert, können weggelassen werden



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

# Längeneinheiten

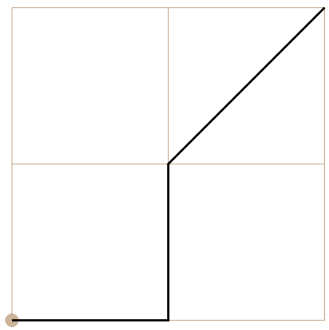
alle gängigen LaTeX-Einheiten sind möglich (cm,mm,pt,ex,em,...)



```
\path[draw]  (10pt,0) -- (4em, 2ex);
```

# relative Koordinaten

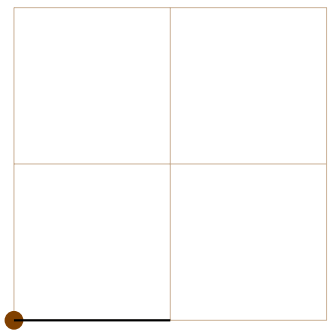
- Variante I: `++(x,y)`
  (echtes Shift: neuer Bezugspunkt);



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

# relative Koordinaten

- Variante I: `++(x,y)`
  (echtes Shift: neuer Bezugspunkt);

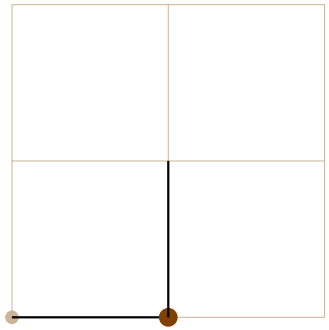


- Bezugspunkt

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

# relative Koordinaten

- Variante I: `++(x,y)`
  (echtes Shift: neuer Bezugspunkt);

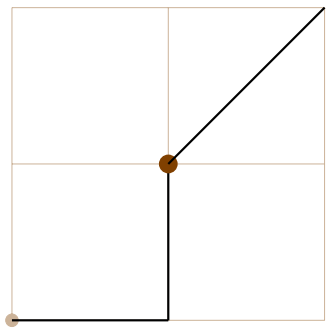


- Bezugspunkt

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

# relative Koordinaten

- Variante I: `++(x,y)`
  (echtes Shift: neuer Bezugspunkt);

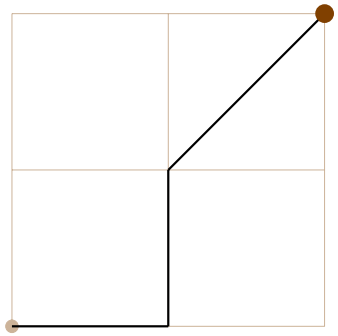


- Bezugspunkt

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

# relative Koordinaten

- Variante I: `++(x,y)`
  (echtes Shift: neuer Bezugspunkt);

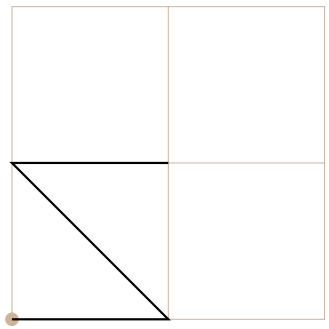


- Bezugspunkt

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

# relative Koordinaten

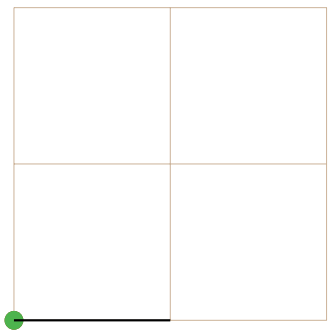
- Variante II: `+(x,y)`
  (temporäres Shift, behalte alten Bezugspunkt)



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

# relative Koordinaten

- Variante II: `+(x,y)`
  (temporäres Shift, behalte alten Bezugspunkt)

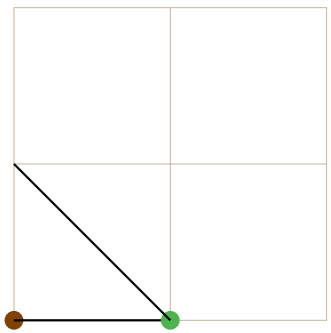


- Bezugspunkt
- vorheriges Pfadende

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

# relative Koordinaten

- Variante II: `+(x,y)`
  (temporäres Shift, behalte alten Bezugspunkt)

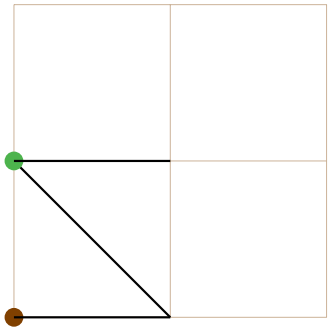


- Bezugspunkt
- vorheriges Pfadende

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

# relative Koordinaten

- Variante II: `+(x,y)`
  (temporäres Shift, behalte alten Bezugspunkt)

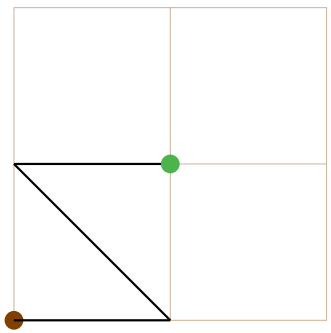


- Bezugspunkt
- vorheriges Pfadende

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

# relative Koordinaten

- Variante II: `+(x,y)`
  (temporäres Shift, behalte alten Bezugspunkt)

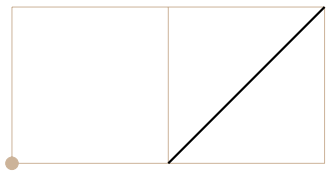


- Bezugspunkt
- vorheriges Pfadende

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

# Pfade

**Beispiel:**



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

- Pfadaktion: `draw`
- Pfadoperation 1: `(1cm,0cm)` “move-to”-Operation
- Pfadoperation 2: `--(1cm,0cm)` “line-to”-Operation

# Pfadoperationen

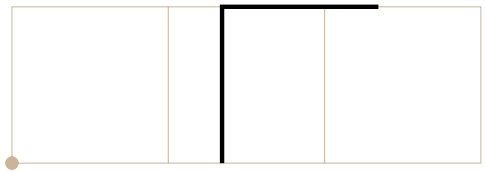
- `coordinate (label)`

```
\path[draw]
(1.344,0) coordinate (A)
  ++(0,1) coordinate (B)
   +(1,0) coordinate (C);
```

- Pfadaktion: `draw` bleibt ohne Effekt (line-to Operation fehlt)
- Pfadoperation `coordinate` vergibt Namen für Punkte

# Pfadoperationen: coordinate

- `coordinate (label)`

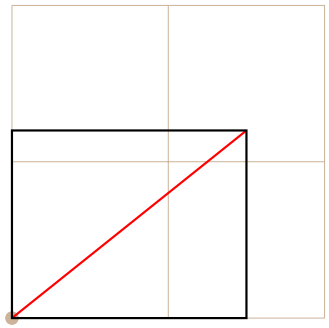


```
\path[draw]
(1.344,0) coordinate (A)
  ++(0,1) coordinate (B)
   +(1,0) coordinate (C);
\path[draw,thick] (A) -- (B) -- (C);
```

- Pfad 2: benutzt diese Namen, `draw` Effekt sichtbar dank line-to Operationen
- Konventionen: Leerzeichen, Zahlen, Buchstaben sind **erlaubt** **nicht erlaubt:** (Doppel-)Punkt, Komma, spezielle Sonderzeichen

# Pfadoperationen: rectangle

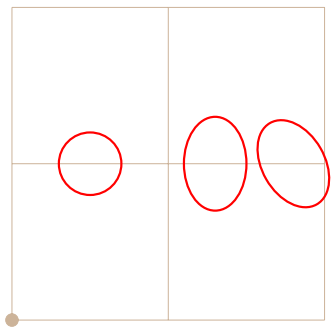
- `(fromNd) rectangle (toNd)`
- `fromNd` und `toNd` sind Koordinaten der Diagonalen



```
\path[draw,red]
(0,0) coordinate (A) --
  +(1.5,1.2) coordinate (B);
\path[draw] (A) rectangle (B) ;
```

# Pfadoperationen: circle

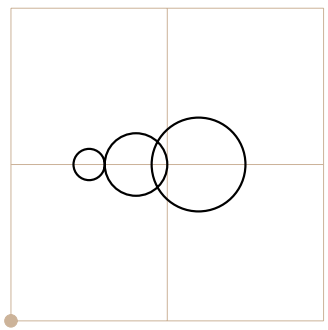
- `(midNd) circle[circle options]`
- `midNd` ist der Mittelpunkt
- Optionen: `radius=<length>`
- Optionen für Ellipsen: `x radius=` , `y radius=` , `rotate=`



```
\path[draw]  (1,1)
 +(-.5,0) circle[radius=0.2cm]
 +(0.3,0) circle[x radius=0.2cm, y radius=0.3cm]
 +(0.8,0) circle
    [x radius=0.2cm, y radius=0.3cm, rotate=30];
```

# Pfadoperationen: `circle`

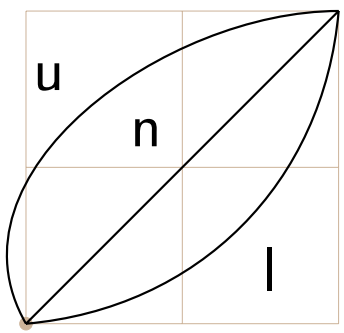
- Notationsclash: alte Notation:
- `(midNd) circle(radius)`
- `midNd` ist der Mittelpunkt
- `radius` wird hier nicht als Option, sondern als "Punkt mit einer Dimension" angegeben
- Fehlerquelle!



```
\path[draw] (1,1)
 ++(-.5,0) circle (0.1cm) %ok
 ++(0.3,0) circle[radius=0.2cm]+( 0.4,0) circle(0.3cm) %ok
 +(1.3,0) circle(2,2) % FEHLER
 +(0.8,0) circle[rotate=20](2cm); % FEHLER
```

# Pfadoperationen: to-Path

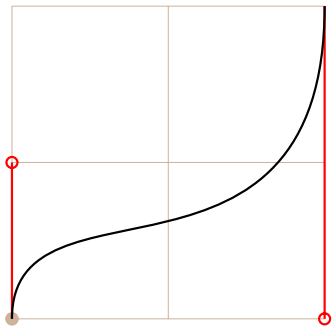
- `(fromNd) to[to options] (toNd)`
- Ohne Optionen äquivalent zu “line-to”-Operation (−−)
- Optionen: `out=`, `in=` : geschwungener Pfad
- Optionen: `bend left=` : geschwungener Pfad
- Optionen: `edge label=` : Beschriftung




```
\path[draw]
(0,0) to[edge label=n] (2,2)
(0,0) to[edge label=u,  out=120, in=180] (2,2)
(0,0) to[edge label'=l, bend right=40] (2,2);
```

# Pfadoperationen: Curve-To

- `(fromNd) .. controls (ctrl1) and (ctrl2) .. (toNd)`
- geschwungener Pfad von `fromNd` nach `toNd`
- Kontrollpunkte `ctrl1` und `ctrl2` bilden Tangenten.

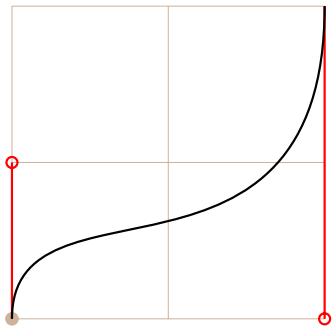


```
\path[draw,red,radius=1pt]
(0,0)  -- (0,1)coordinate(ctrl1)circle[]
(2,0)coordinate(ctrl2) circle--(2,2);
\path[draw](0,0)..controls(ctrl1)and(ctrl2)..(2,2);
```

- GIMP - Bsp.

# Pfadoperationen: Curve-To

- `(fromNd) .. controls (ctrl1) and (ctrl2) .. (toNd)`
- geschwungener Pfad von `fromNd` nach `toNd`
- Kontrollpunkte `ctrl1` und `ctrl2` bilden Tangenten.



```
\path[draw,red,radius=1pt]
(0,0)  -- (0,1)coordinate(ctrl1)circle[]
(2,0)coordinate(ctrl2) circle--(2,2);
\path[draw](0,0)..controls(ctrl1)and(ctrl2)..(2,2);
```

- GIMP - Bsp.

# Pfadoperationen: viele viele mehr . . .

- Rounding corners
- Arc Operations (Kreisbögen)
- Grid Operations (Gitter)
- Parabola, Sine, Cosine
- Plot Operation (Graphen von allg. Funktionen)
- Horizontal and Vertical Lines (Manhatten-Style)
- Node- and Edge Operationen (Später!)
- . . .

# Tagesprogramm

1. TikZ I - Einleitung
2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

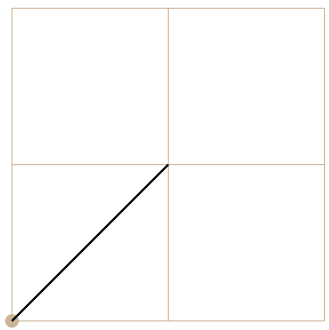
# Pfadaktionen

- `\path[draw]`, kurz `\draw`
- `\path[fill]`, kurz `\fill`
- `\path[shade]`, kurz `\shade`
- `\path[clip]`, kurz `\clip`
- Decorations

## Kombinationen Möglich

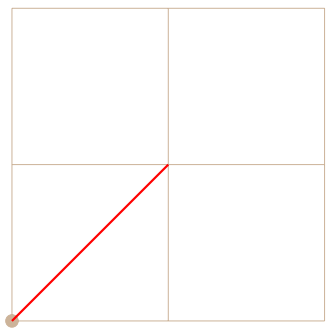
- `\path[fill,draw]`, kurz `\filldraw`
- `\path[draw,shade]`, kurz `\shadedraw`

# Pfadaktionen: draw



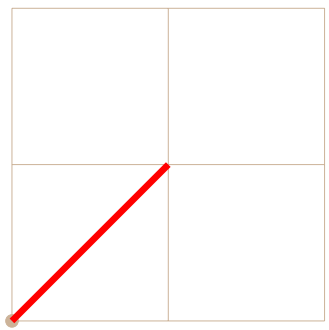
```
\path[draw]
(0,0)--(1,1);
```

# Pfadaktionen: draw



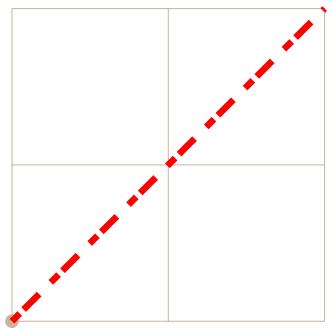
```
\path[draw=red]
(0,0)--(1,1);
```

# Pfadaktionen: draw



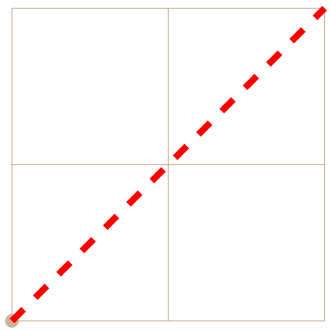
```
\path[draw=red,very thick]
(0,0)--(1,1);
```

# Pfadaktionen: draw



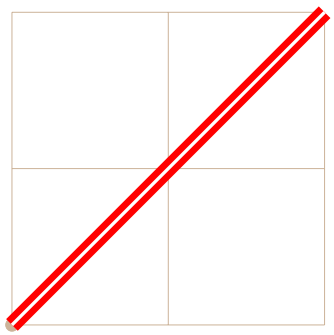
```
\path[draw=red,very thick
, dash pattern=on 2pt off 1pt on 4pt off 3pt]
(0,0)--(2,2);
```

# Pfadaktionen: draw



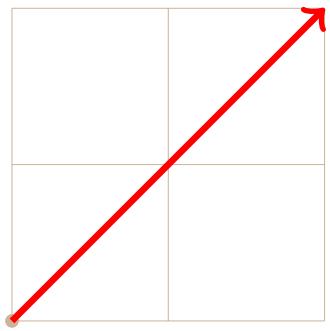
```
\path[draw=red,very thick
, dashed]
(0,0)--(2,2);
```

# Pfadaktionen: draw



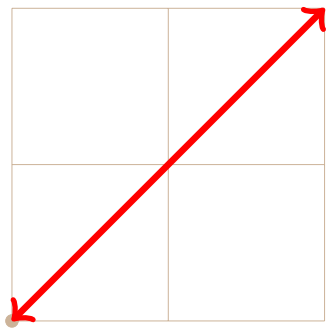
```
\path[draw=red,very thick
, double]
(0,0)--(2,2);
```

# Pfadaktionen: draw



```
\path[draw=red,very thick
, ->]
(0,0)--(2,2);
```

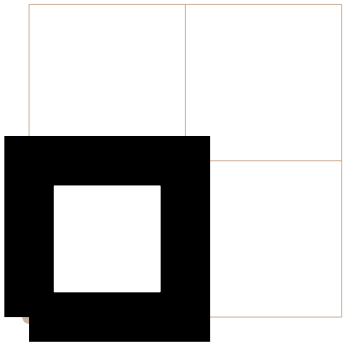
# Pfadaktionen: draw



```
\path[draw=red,very thick
, <->]
(0,0)--(2,2);
```

# Pfadaktionen: `draw`

Achtung: Manche Effekte sind subtil!

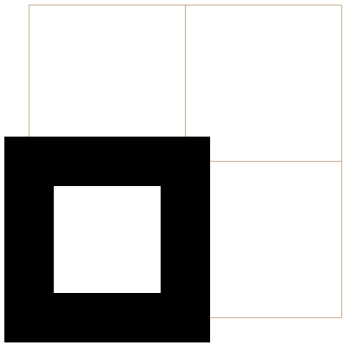


```
\draw[line width=9pt]
(0,0)--(1,0)--(1,1)--(0,1)--(0,0);
```

Abhilfe: entweder `rectangle` zeichnen, oder `cycle` Anweisung

# Pfadaktionen: draw

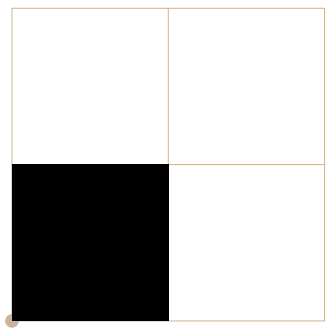
Achtung: Manche Effekte sind subtil!



```
\draw[line width=9pt]
(0,0)--(1,0)--(1,1)--(0,1)--cycle;
```

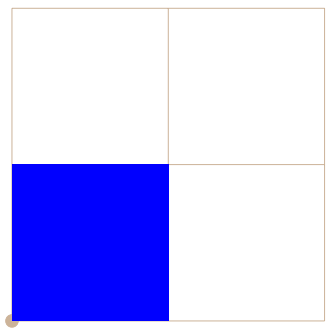
- `cycle` schließt den Pfad in der Anfangsrichtung

# Pfadaktionen: `fill`



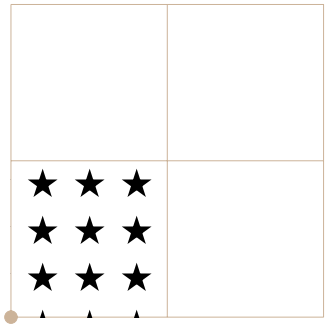
```
\path[fill]
(0,0)--(0,1)--(1,1)--(1,0)--(0,0);
```

# Pfadaktionen: `fill`



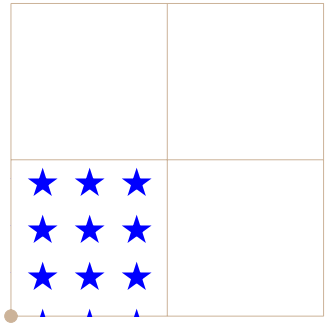
```
\path[fill=blue]
(0,0)--(0,1)--(1,1)--(1,0)--(0,0);
```

# Pfadaktionen: `fill`



```
\path[fill=blue, , pattern=fivepointed stars]
(0,0)--(0,1)--(1,1)--(1,0)--(0,0);
```

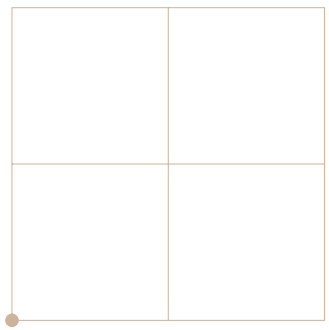
# Pfadaktionen: `fill`



```
\path[pattern=fivepointed stars, pattern color = blue]
(0,0)--(0,1)--(1,1)--(1,0)--(0,0);
```

# Pfadaktionen: `fill`

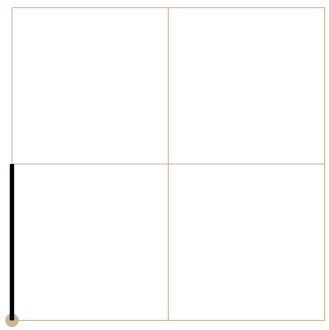
- Vorsicht: Vehalten, bei ungeschlossenen Pfaden!



```
\filldraw [thick]
(0,0)  -- (0,1)--(1.5,1)--(1,2);
```

# Pfadaktionen: `fill`

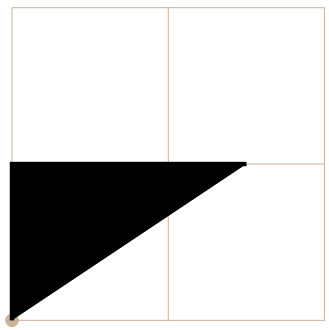
- Vorsicht: Vehalten, bei ungeschlossenen Pfaden!



```
\filldraw [thick]
(0,0)  -- (0,1)--(1.5,1)--(1,2);
```

# Pfadaktionen: `fill`

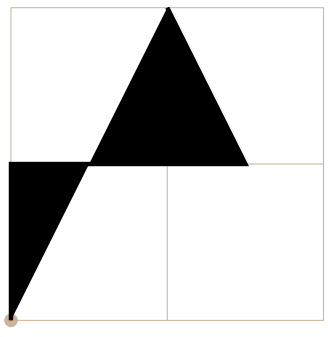
- Vorsicht: Vehalten, bei ungeschlossenen Pfaden!



```
\filldraw [thick]
(0,0)  -- (0,1)--(1.5,1)--(1,2);
```

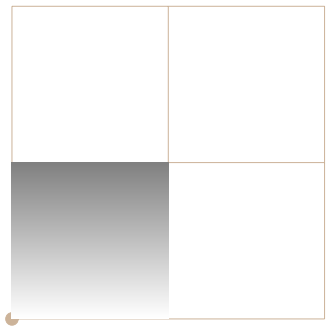
# Pfadaktionen: `fill`

- Vorsicht: Vehalten, bei ungeschlossenen Pfaden!



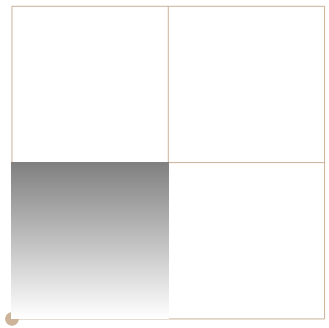
```
\filldraw [thick]
(0,0)  -- (0,1)--(1.5,1)--(1,2);
```

# Pfadaktionen: shade



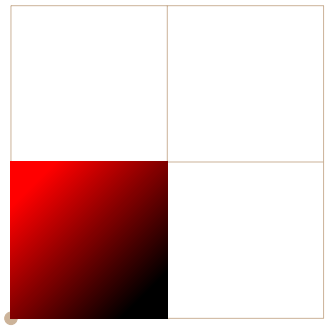
```
\path[shade]
(0,0)rectangle(1,1);
```

# Pfadaktionen: shade



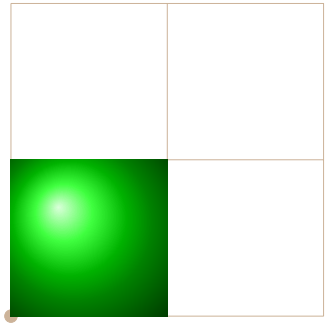
```
\path[shade, shading=axis]
(0,0)rectangle(1,1);
```

# Pfadaktionen: shade



```
\path[shade, shading=axis, top color=red,
bottom color=black
, shading angle=45]
(0,0)rectangle(1,1);
```

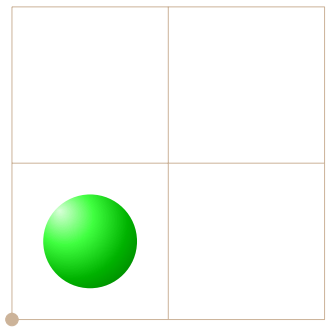
# Pfadaktionen: shade



```
\path[shade, shading=ball, ball color=green]
(0,0)rectangle(1,1);
```

# Pfadaktionen: clip

- Clipping definiert eine Art Bilderrahmen/ Schablone

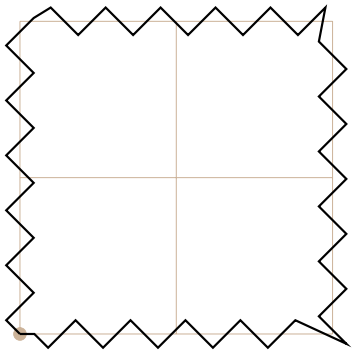


```
\path[clip]
(.5,.5)circle[radius=0.3];
 \path[shade, shading=ball, ball color=green]
(0,0)rectangle(1,1);
```

- Clipping muss *vor* dem Rest des Bildes (Scopes) passieren.
- wird erst mit „Scopes“ wirklich nützlich (später)

# Pfadaktionen: decorate

- Dekorationen wandeln orignalen Pfad um, oder ersetzen diesen

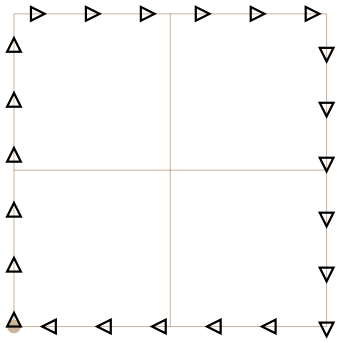


```
\path[draw, decorate, decoration= zigzag]
(0,0)rectangle(2,2);
```

- Pfad wurde umgewandelt (`decorations.pathmorphing`)
- benötigt `decorations.pathmorphing` tikz-Bibliothek

# Pfadaktionen: decorate

- Dekorationen wandeln orignalen Pfad um, oder ersetzen diesen



```
\path[draw, decorate, decoration= triangles]
(0,0)rectangle(2,2);
```

- Pfad wurde durch Teilpfade ersetzt (`decorations.pathreplacing`) bzw. durch Shapes ersetzt (`decorations.shapes`)
- Pfad lässt sich so nicht mehr füllen (extra `\fill`-Pfad nötig)

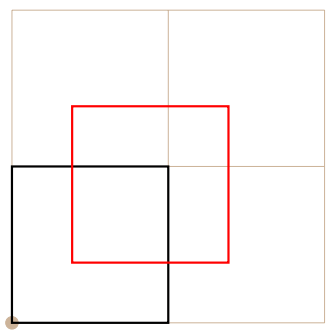
# Tagesprogramm

1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

# Standardkoordinatensystem

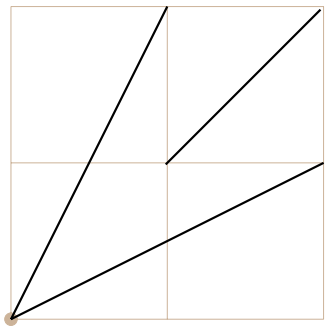
- Euklidisch, 3D
- Einheitsvektoren der Länge 1cm
- fehlende z-Koordinate wird auf Null gesetzt



```
\draw (0,0,0)rectangle(1,1);
\draw[red] (0,0,-1)rectangle(1,1,-1);
```

# Koordinatenransformation

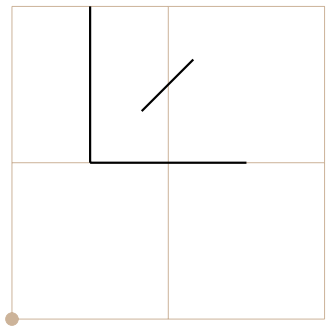
- lineare Transformation
- (Re-)Definition der Einheitsvektoren
- lokale Koordinatentransformation im Pfad



```
\path[draw, x={(1cm,2cm)}, y={(2cm,1cm)}]
(0,0)--(1,0)
(0,0)--(0,1)
(.33,.33)--(.66,.66);
```

# Koordinatenransformation

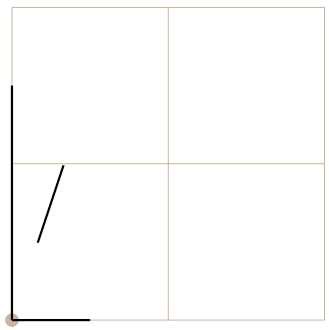
- Translation
- lokale Koordinatentransformation im Pfad



```
\path[draw, shift={(0.5cm,1cm)}]
(0,0)--(1,0)
(0,0)--(0,1)
(.33,.33)--(.66,.66);
```

# Koordinatentransformation

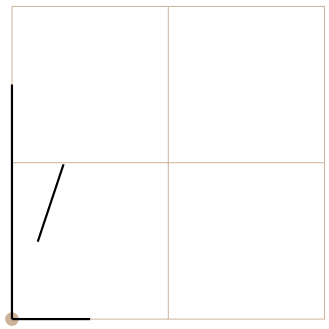
- Skalierung
- lokale Koordinatentransformation im Pfad



```
\path[draw, xscale=0.5, yscale=1.5]
(0,0)--(1,0)
(0,0)--(0,1)
(.33,.33)--(.66,.66);
```

# Koordinatentransformation

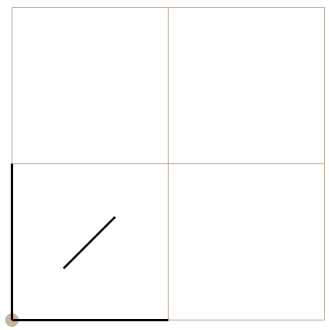
- Skalierung
- lokale Koordinatentransformation im Pfad



```
\path[draw, xscale=0.5, yscale=1.5]
(0,0)--(1,0)
(0,0)--(0,1)
(.33,.33)--(.66,.66);
```

# Koordinatentransformation

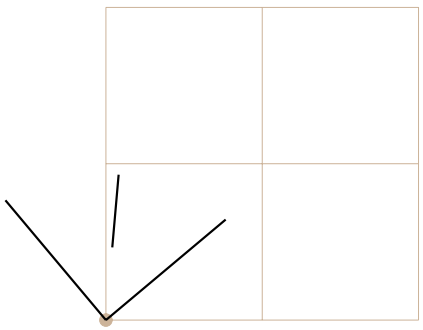
- Rotation `rotate=<Winkel>`
- lokale Koordinatentransformation im Pfad



```
\path[draw]
(0,0)--(1,0)
(0,0)--(0,1)
(.33,.33)--(.66,.66);
```

# Koordinatentransformation

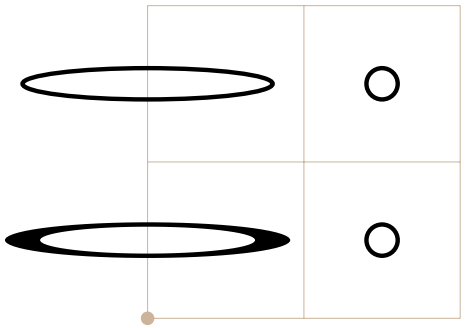
- Rotation `rotate=<Winkel>`
- lokale Koordinatentransformation im Pfad



```
\path[draw, rotate=40]
(0,0)--(1,0)
(0,0)--(0,1)
(.33,.33)--(.66,.66);
```

# Koordinatentransformation

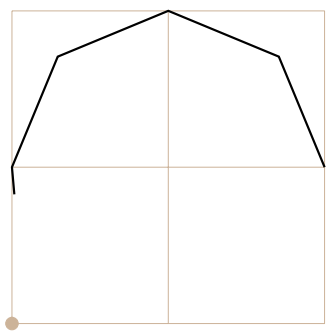
**Wichtig:** Es werden wirklich nur Koordinaten transformiert: Vgl.:



```
\draw[thick](1.5,1.5)circle(1mm)
            (1.5,0.5)circle(1mm);
 \draw[thick,xscale=8](0,1.5) circle(1mm);
 \pgflowlevel{\pgftransformxscale{8}}
 \draw[thick](0,0.5) circle(1mm);
```

# Polarkoordinaten

- Punkt wird spezifizert in der From `(<Winkel>:<Radius)`
- `<Winkel>` bezogen auf Koordinatenursprung und positive $x$-Achse; entgegen dem Uhrzeigersinn (math. positiv)
- `<Radius>` vorgeschrieben: Angabe *mit* Längeneinheit



```
\draw (1,1) +(0: 1cm)  --
            +(45: 1cm) --
            +(90: 1cm) --
            +(135: 1cm) --
            +(180: 1cm)--
            +(190: 1cm);
```

# Einschub: `foreach` Schleife

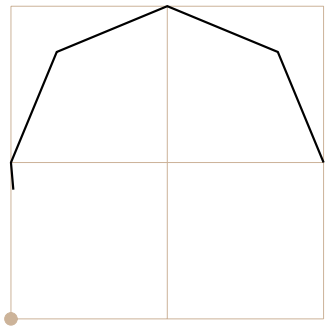
- Im Pakte `pgffor` enthalten, wird von TikZ automatisch geladen
- kann auch unabhängig geladen werden: `\usepackage{pgffor}`

Syntax:

```
\foreach \<iterName in> { <valuesI>, <valuesII>, ... }
        {  <commands>  }
```

# Einschub: foreach Schleife

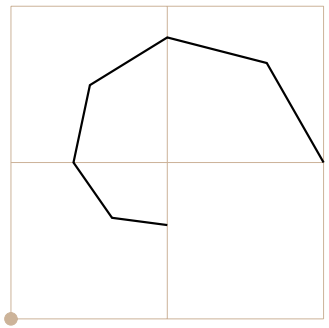
Bsp. von Eben:



```
\draw (1,1) +(0:1cm)
\foreach \ang in {45, 90, 135, 180, 190}
{ -- +( \ang : 1cm)}   ;
```

# Einschub: `foreach` Schleife

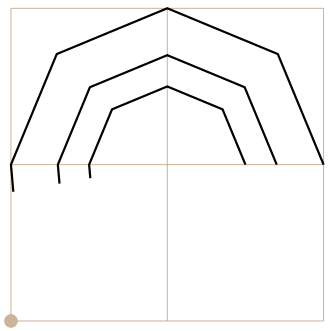
Auch Iteration über Tupel ist möglich:



```
\foreach \ang/\rad in {45/0.9cm, 90/0.8cm, 135/0.7cm,
180/0.6cm, 225/0.5cm, 270/0.4cm}
{ -- +( \ang : \rad)}   ;
```

# Einschub: foreach Schleife

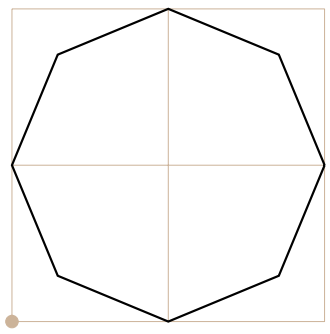
Auch Schachtelung möglich:



```
\draw (1,1) \foreach \rad  in {1cm, 0.7cm, 0.5cm}
    { +(0:\rad) \foreach \ang in {45, 90, 135, 180, 190}
                  {   -- +( \ang : \rad)}%end angle-Loop
    }  ;    %end radius-Loop
```

# Einschub: `foreach` Schleife

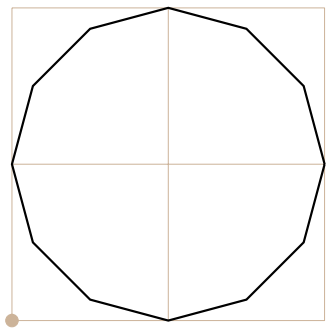
kürzere Schreibweise:



```
\foreach \ang in {45, 90, ..., 360}
{ -- +( \ang : 1cm)}  ;
```

# Einschub: `foreach` Schleife

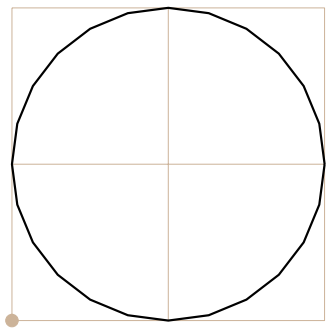
kürzere Schreibweise:



```
\foreach \ang in {30, 60, ..., 360}
{ -- +( \ang : 1cm)}  ;
```

# Einschub: `foreach` Schleife

kürzere Schreibweise:



```
\foreach \ang in {15, 30, ..., 360}
{ -- +( \ang : 1cm)}  ;
```

# Tagesprogramm

1. TikZ I - Einleitung

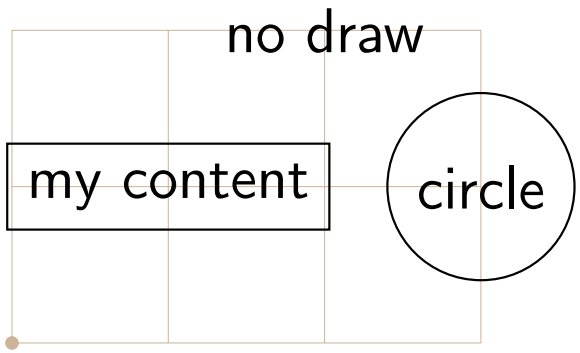
2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

# Konzept: Knoten

Ein Knoten (TikZ-node)

- wird mit der Pfadoperation `node` erzeugt
- hat einen Inhalt, welcher in eine LaTeX-Box gesetzt wird (default= `\hbox`)
- begrenzende Form passt sich dem Inhalt an.
- umfasst (meist) drei Konzepte:
  - eine äußere Form (z.B. Rechteck, Kreis, ...)
  - einen Inhalt (z.B. Text), und geeignete LaTeX-Box
  - einen symbolischen Namen (wie bei `coordinate`-Pfadoperation)
- man muss nicht alle Konzepte gleichzeitig benutzen

# Knoten, Bsp.




```
\draw (2,2) node[rectangle]  {no draw};
\draw (1,1) node[draw,rectangle] (myname) {my content};
\draw (3,1) node[draw,circle] (myname2) {circle};
```

# Tagesprogramm

1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

# Konzept: Knoten

Eine mögliche Syntax:

`node[<Optionen>] (<Name>){Inhalt}`

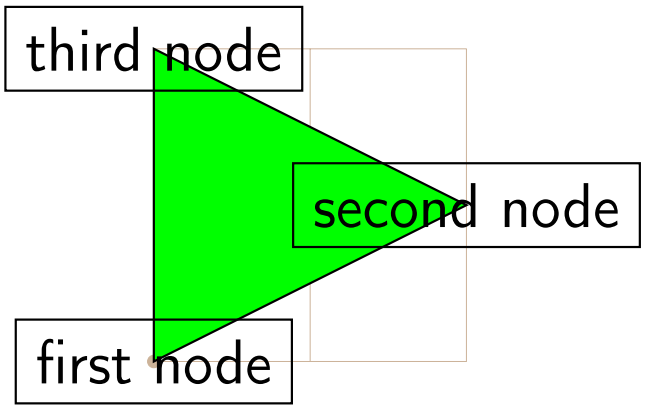
**Achtung!:**

- `(<Name>)`, sowie `[<Optionen>]` sind optional
- es **muss** aber ein Inhalt spezifiziert werden (`{}` darf nicht fehlen), dieser darf aber leer bleiben. (Vgl. `coordinate`: hier wird niemals ein Inhalt angegeben).
- Optionen für
  - Form des Knotens ( `rectangle, circle,...`)
  - ob und wie der Knoten(rahmen) gezeichnet, gefüllt, ... wird
  - welcher *Anker* des Knotens als Referenzpunkt dient

Auf Knoten gehen wir noch genauer ein (TikZ II > Januar)

# Konzept: Knoten

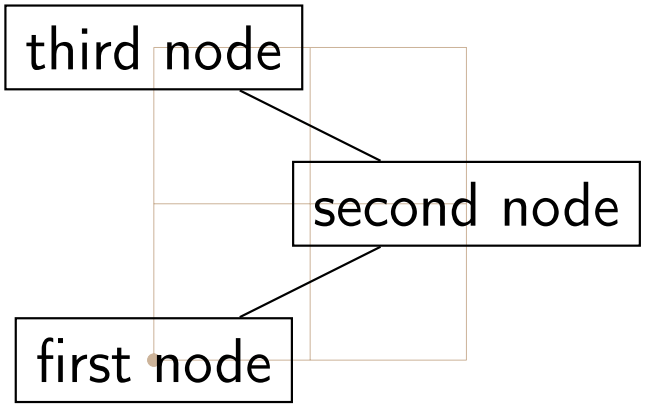
Wo es hinführt?




```
\filldraw[fill=green]
(0,0) node[draw] {first node}
-- (2,1) node[draw] {second node}
-- (0,2) node[draw] {third node} -- cycle;
```

# Konzept: Knoten

Wo es hinführt?




```
\path (0,0) node[draw] (A) {first node}
 (2,1) node (B)[draw] {second node}
 (0,2) node (C)[draw] {third node};
\filldraw[fill=green]
 (A)--(B)--(C) -- cycle;
```

# Einschub: Farben, Transparanz

`color` *rightarrow* `xcolor` *rightarrow* `xxcolor` wird automatisch geladen.

**Features:**

- verschiedene Farbmodelle (`rgb`, `cmyk`, `grayscale`, `...`)
- verschiedene Farbpaletten, und Namensgebungen einstellbar
- Eigene Farbdefinitionen und Aliasse
- Transparenz ist dagegen Bestandteil von `pgf`

**Hier:** nur einige Beispiele

# Einschub: Farben

## Vordefinierte Grundarben

black darkgray lime pink violet blue gray magenta purple white brown green olive red yellow cyan lightgray orange teal

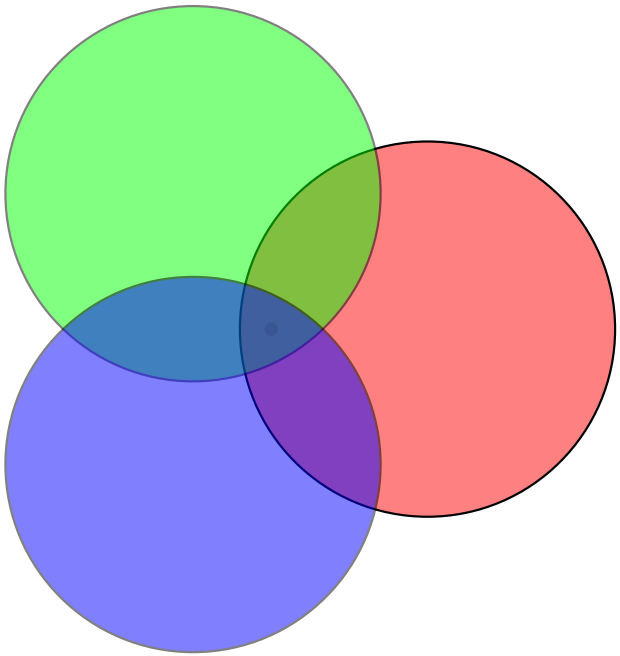
## Definition eigener Farben

```
\definecolor{hellesbrombeer}{rgb}{0.8,0.4,0.8}
```

- Erweiterte Farbangaben: `\colorbox{red!10!white}` 10 % rot, Rest weiß
- Aliasse: `\colorlet{myCol}{red!10!white}` `\colorbox{myCol}{this is myCol}` this is myCol

# Einschub: Transparanz

- `draw opacity=..`, `fill opacity=..`, `opacity=..`



```
\filldraw[fill=red,fill opacity=0.5]
(0:1cm) circle (12mm);
\filldraw[fill=green,opacity=0.5]
(120:1cm) circle (12mm);
\filldraw[fill=blue,opacity=0.5]
(-120:1cm) circle (12mm);
```

# Tagesprogramm

1. TikZ I - Einleitung

2. Grundlegende Konzepte
   - Pfade und Koordinaten
   - Pfade, Pfadaktionen, Pfadoperationen
   - Pfadaktionen
   - Koordinatensysteme
   - Knoten
   - Knoten
   - Möglichkeiten der Strukturierung

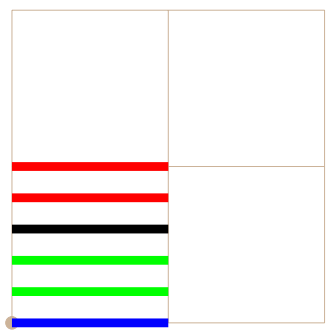
# Eigene Kommandos

- Mehr denn je: Eigene Kommandos !
- Evtl. sogar: Eigene Umgebungen
- Keine Teilbilder doppelt zeichnen ! (Siehe auch HA)

# Konzept: Scopes

- `Scope`s sind Umgebungen in TikZ-Bildern, sie definieren einen Geltungsbereich
- Syntax: `\begin{scope}[options]`
  `<Pfade> \end{scope}`
- (Fast) alle Pfadoptionen, Koordinatentransformationen, . . . , ect. können auch in `Scope`s angegeben werden.
- Sie werden auf dann **alle** Pfade im `Scope` angewendet, falls dies möglich/sinnvoll ist, Bsp.:
- Clipping Operationen bleiben auf `Scope`s beschränkt
- Die optionalen Argumente der `tikzpicture`-Umgebung bilden den äußersten (Standard-)`Scope`

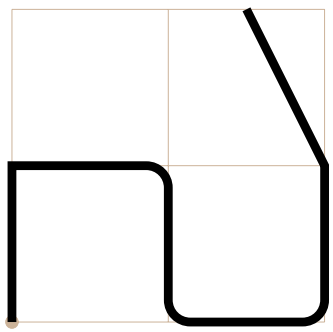
# Konzept: Scopes



```
\begin{scope}[ultra thick]
\begin{scope}[draw=red]
\draw  (0,1) -- (1,1);
\draw (0,0.8) -- (1,0.8);
\end{scope}
\draw  (0,0.6) -- (1,0.6);
\begin{scope}[green]
\draw (0,0.4) -- (1,0.4);
\draw (0,0.2) -- (1,0.2);
\draw[blue] (0,0) -- (1,0);
\end{scope}
\end{scope}
```

# Scope, geht auch im Pfad

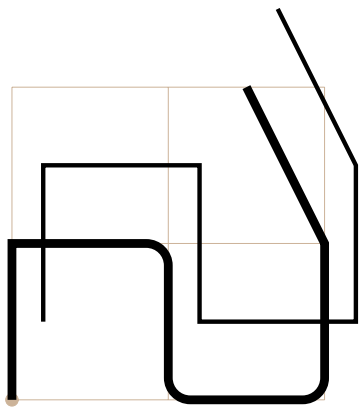
- Syntax, ähnlich zu Scopes in LaTeX
- öffnen: `{` , schließen `}` , Optionen `[<options>]`
- dürfen nur nach vollendeter Pfadoperation beginnen o. enden.
- viele Pfadoptionen lassen sich nur auf *gesamten* Pfad anwenden (Farbe, Linenart, u.a.)



```
\draw[ultra thick] (0,0)--(0,1)
{[rounded corners]--(1,1)
--(1,0)--(2,0)}
--(2,1)--(1.5,2);
```

# Scope, geht auch im Pfad

- Syntax, ähnlich zu Scopes in LaTeX
- öffnen: `{` , schließen `}` , Optionen `[<options>]`
- dürfen nur nach vollendeter Pfadoperation beginnen o. enden.
- viele Pfadoptionen lassen sich nur auf *gesamten* Pfad anwenden (Farbe, Linenart, u.a.)

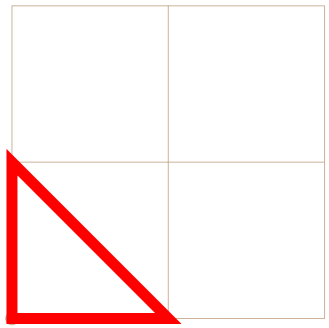


```
\draw[ultra thick] (0,0)--(0,1)
{[rounded corners]--(1,1)
--(1,0)--(2,0)}
--(2,1)--(1.5,2);
```

```
\draw[thick,yshift=0.5cm,xshift=0.2cm] (0,0)--(0,1)
{[red]--(1,1) --(1,0)--(2,0)}
--(2,1)--(1.5,2); % Scope(red) ohne Effekt
```

# Konzept: TikZ-Keys und Styles

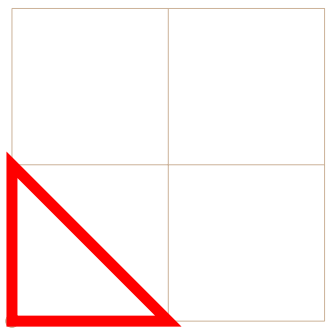
- Key-Value Syntax: Alle bisherigen Optionen lassen sich einem *Key* zuordnen
- um das Rendern zu beeinflussen, werden für diese *Key*s Werte (*Values*) gesetzt
- Man mus wissen:
  - Welche Keys es gibt
  - Welche Values zulässig sind
  - Wie diese Kombination das Bild beeinflusst



```
\draw[line width=2pt,color=red]
(1,0) -- (0,0) -- (0,1) -- cycle;
```

# Konzept: TikZ-Keys und Styles

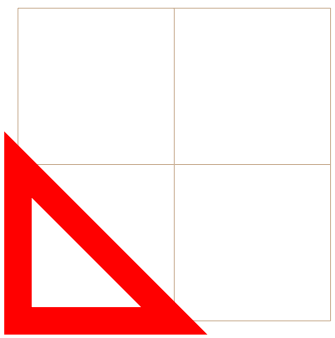
- Idee: speichere häufig verwendete Kombinationen
- Syntax:
  `\tikzset{<Style-Name>/.style={<Style-Optionen>}`



```
\tikzset{my red line/.style={line width=2pt, color=red}}
\draw[my red line] (1,0) -- (0,0) -- (0,1) -- cycle;
```

# Konzept: TikZ-Keys und Styles

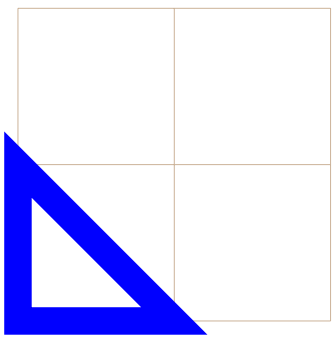
- Idee: speichere häufig verwendete Kombinationen
- Syntax:
  `\tikzset{<Style-Name>/.style={<Style-Optionen>}}`
- das geht auch Parametrisiert



```
\tikzset{my red line/.style={line width=#1, color=red}}
\draw[my red line=5pt] (1,0) -- (0,0) -- (0,1) -- cycle;
```

# Konzept: TikZ-Keys und Styles

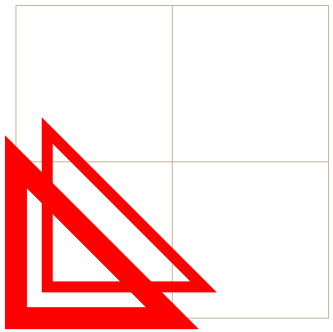
- Idee: speichere häufig verwendete Kombinationen
- Syntax:
  `\tikzset{<Style-Name>/.style={<Style-Optionen>}}`
- das geht auch Parametrisiert, mit mehr als 2 Parametern



```
\tikzset{my line/.style 2 args={line width= #1, color= #2}}
\draw[my line={5pt}{blue}]
(1,0) -- (0,0) -- (0,1) -- cycle;
```

# Konzept: TikZ-Keys und Styles

- Idee: speichere häufig verwendete Kombinationen
- Shortcut: `\tikzstyle{<style-name>}=[<style-options>]`
- Allerdings (bisher?) maximal ein Argument und globale Definition



```
\tikzstyle{my red line}[2pt]=[line width=#1, color=red]
\draw[my red line=4pt] (1,0) -- (0,0) -- (0,1) -- cycle;

\draw[shift={(0.2cm,0.2cm)}, my red line]
(1,0) -- (0,0) -- (0,1) -- cycle;
```

# 3-Sat Reduktion

# Outlook TikZ II

- mehr zu TikZ-`nodes`
- TikZ-`edges`, `matrix`, `graph`
- Plots mit `pgfplots` und TikZ
- Interaktion mit LaTeX-Beamer
- evtl. externalize

# Outlook TikZ II

- mehr zu TikZ-`nodes`
- TikZ-`edges`, `matrix`, `graph`
- Plots mit `pgfplots` und TikZ
- Interaktion mit LaTeX-Beamer
- evtl. externalize

Vielen Dank !

# Outlook TikZ II

- mehr zu TikZ-`nodes`
- TikZ-`edges`, `matrix`, `graph`
- Plots mit `pgfplots` und TikZ
- Interaktion mit LaTeX-Beamer
- evtl. externalize

Vielen Dank !

(RB-Zeit entfällt heute wegen Krankheit)