Beamer and TikZ Workshop

Andrew Mertz
William Slough

Mathematics and Computer Science Department
Eastern Illinois University

July 17, 2007

Afternoon overview: Ti*k*Z

- One of many drawing choices found in the TEX jungle
- PGF is the "back end" — Ti*k*Z provides a convenient interface
- Ti*k*Z shares some ideas with MetaPost and PSTricks
- Multiple input formats: plain TEX, LATEX
- Multiple output formats: PDF, PostScript, HTML/SVG
- Works well with beamer and incremental slides
- Extensive documentation (405 pages)

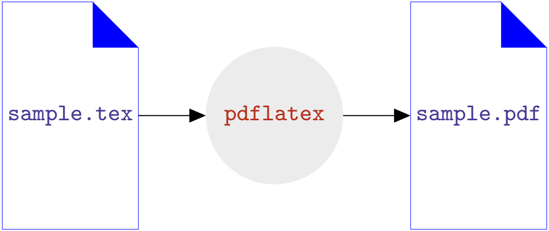
Ti*k*Z capabilities: a laundry list

- 2-d points
- 3-d points
- Cartesian coordinates
- Polar coordinates
- Relative positions
- Named points
- Line segments
- Bézier curves
- Rectangles
- Circles
- Ellipses
- Arcs
- Grids
- Text
- Drawing
- Clipping
- Filling
- Shading
- Iteration
- Transformations
- Libraries

Layout of a Ti*k*Z-based document using LaTeX

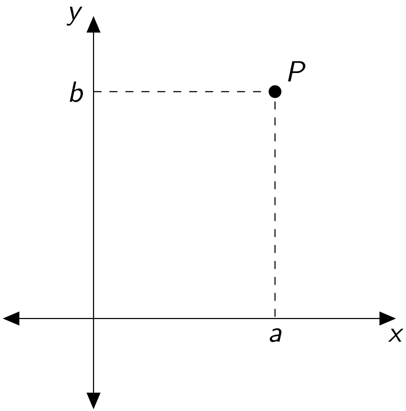
```
\documentclass[11pt]{article}
...
\usepackage{tikz}

% Optional pgf libraries
\usepackage{pgflibraryarrows}
\usepackage{pgflibrarysnakes}
...

\begin{document}
   ...
   \begin{tikzpicture}
      ...
   \end{tikzpicture}
   ...
\end{document}
```

Processing a Ti*k*Z-based document



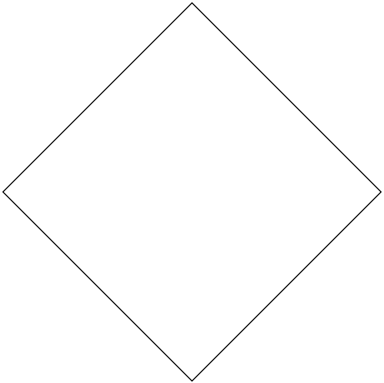
- Text and diagrams can be combined in one file
- Stand-alone graphics can be obtained with `pdfcrop`

2-d points



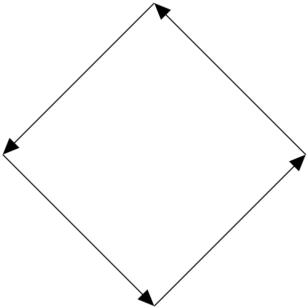
```
\path (a, b) coordinate (P);
```

A diamond of 2-d points



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

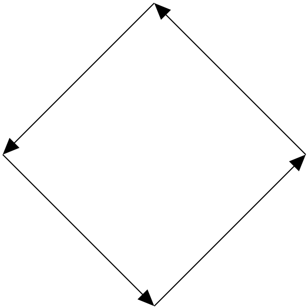
A diamond of 2-d points, with arrows



```
\begin{tikzpicture}[>=triangle 45]
  \draw[->] ( 1, 0) -- ( 0, 1);
  \draw[->] ( 0, 1) -- (-1, 0);
  \draw[->] (-1, 0) -- ( 0,-1);
  \draw[->] ( 0,-1) -- ( 1, 0);
\end{tikzpicture}
```

- >= sets arrow type
- ->, <-, and <->, sets where arrowheads are to be placed
- Arrows will be placed only on the last part of a path

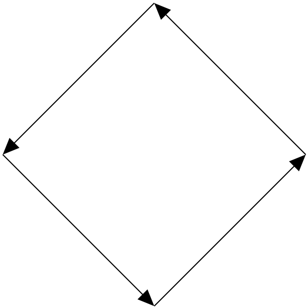
A diamond of 2-d points, with arrows



```
\begin{tikzpicture}[>=triangle 45]
  \draw[->] ( 1, 0) -- ( 0, 1);
  \draw[->] ( 0, 1) -- (-1, 0);
  \draw[->] (-1, 0) -- ( 0,-1);
  \draw[->] ( 0,-1) -- ( 1, 0);
\end{tikzpicture}
```

- >= sets arrow type
- ->, <-, and <->, sets where arrowheads are to be placed
- Arrows will be placed only on the last part of a path

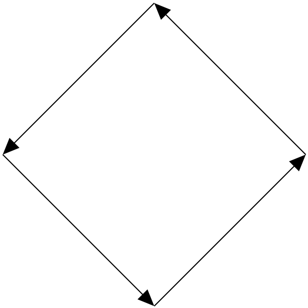
A diamond of 2-d points, with arrows



```
\begin{tikzpicture}[>=triangle 45]
  \draw[->] ( 1, 0) -- ( 0, 1);
  \draw[->] ( 0, 1) -- (-1, 0);
  \draw[->] (-1, 0) -- ( 0,-1);
  \draw[->] ( 0,-1) -- ( 1, 0);
\end{tikzpicture}
```

- >= sets arrow type
- ->, <-, and <->, sets where arrowheads are to be placed
- Arrows will be placed only on the last part of a path

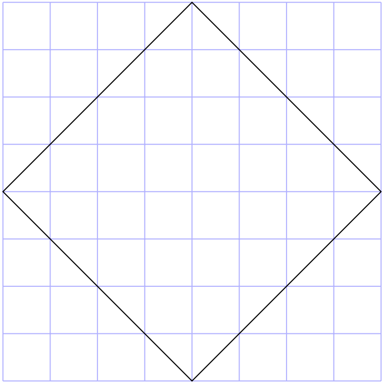
A diamond of 2-d points, with arrows



```
\begin{tikzpicture}[>=triangle 45]
  \draw[->] ( 1, 0) -- ( 0, 1);
  \draw[->] ( 0, 1) -- (-1, 0);
  \draw[->] (-1, 0) -- ( 0,-1);
  \draw[->] ( 0,-1) -- ( 1, 0);
\end{tikzpicture}
```

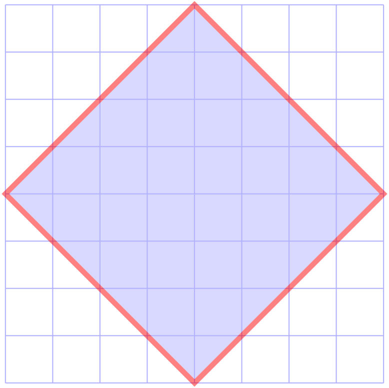
- >= sets arrow type
- ->, <-, and <->, sets where arrowheads are to be placed
- Arrows will be placed only on the last part of a path

A diamond of 2-d points, with grid



```
\draw[step=0.25cm,color=blue!30] (-1,-1) grid (1,1);
\draw (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
```

A few path options

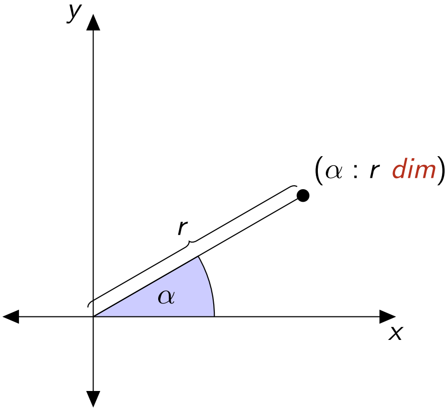


- Fill path
- Change pen color
- Change pen width
- Adjust transparency

A few path options: details

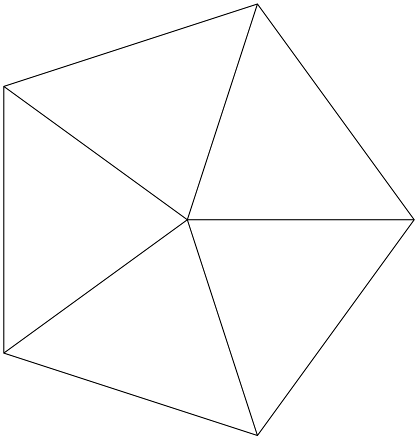
```
\begin{tikzpicture}
   \draw[step=0.25cm,color=blue!30]
         (-1,-1) grid (1,1);
   \draw[fill,color=blue!30,draw=red,
         line width=2pt,opacity=0.5]
         (1,0) -- (0,1) -- (-1,0) -- (0,-1) -- cycle;
\end{tikzpicture}
```

Points specified with polar coordinates



dim: A dimensional unit (cm, pt, mm, in, . . .)

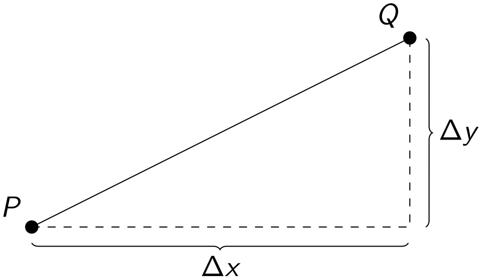
Specifying points with named coordinates



Specifying points with named coordinates: details

```
\begin{tikzpicture}
    \path (0,0) coordinate (origin);
    \path (0:1cm) coordinate (P0);
    \path (1*72:1cm) coordinate (P1);
    \path (2*72:1cm) coordinate (P2);
    \path (3*72:1cm) coordinate (P3);
    \path (4*72:1cm) coordinate (P4);

    % Pentagon edges
    \draw (P0) -- (P1) -- (P2) -- (P3) -- (P4) -- cycle;

    % Spokes
    \draw (origin) -- (P0)  (origin) -- (P1)
          (origin) -- (P2)  (origin) -- (P3)
          (origin) -- (P4);
  \end{tikzpicture}
```

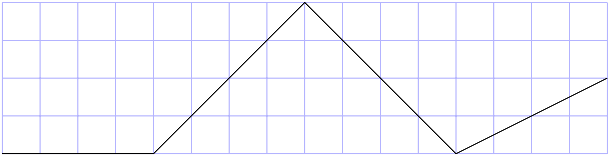
Specifying points with relative coordinates



Given coordinate P and offsets Δx and Δy, we can establish Q:

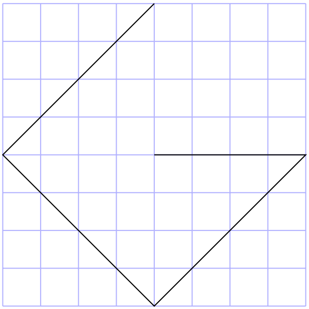
```
\path (P) ++ (Δx,Δy) coordinate (Q);
```

Example with relative offsets



```
\draw (0,0)    --
      ++(1,0)  --
      ++(1,1)  --
      ++(1,-1) --
      ++(1,0.5);
```

Example with relative offsets: ++ vs. +

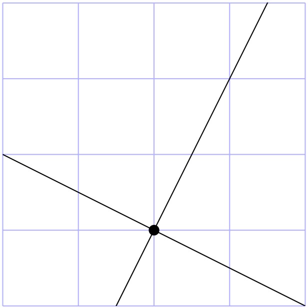


```
\draw (0,0)   --
      +(1,0)  --
      +(0,-1) --
      +(-1,0) --
      +(0,1);
```

++ specifies a point and updates the current location

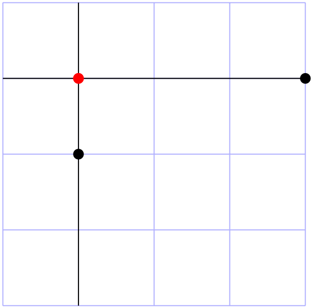
+ only specifies a point

Specifying points as intersections



```
\fill (intersection of 0,2--4,0 and 0,-3--4,5) circle (2pt);
```

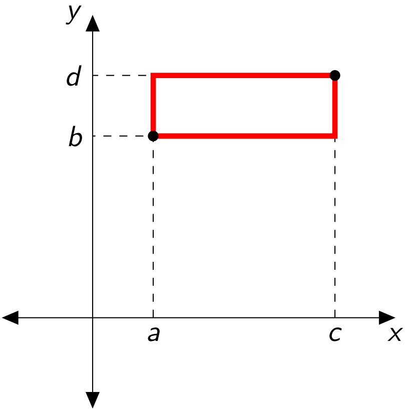
Intersections of horizontal and vertical lines



```
\fill[red] (1,2 |- 4,3) circle (2pt);
\fill[red] (4,3 -| 1,2) circle (2pt);
```

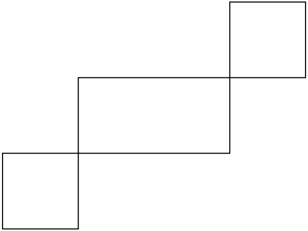
Both `(p |- q)` and `(q -| p)` specify the intersection of the vertical line through `p` and the horizontal line through `q`.

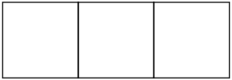
Rectangles



```
\draw (a, b) rectangle (c, d);
```

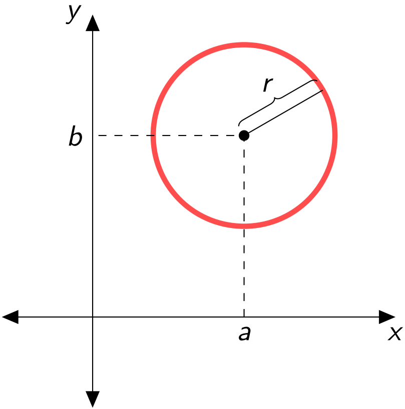
Rectangles: examples





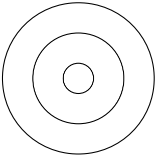
```
\draw (0,0) rectangle (1,1)
            rectangle (3,2)
            rectangle (4,3);

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

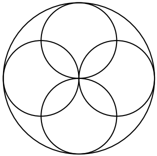
Circles



\draw (*a*, *b*) circle (*r dim*);

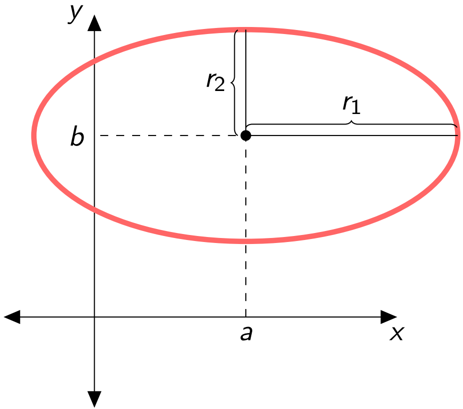
Circles: examples





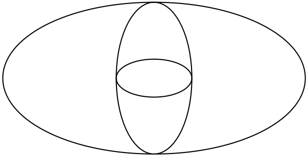
```
\draw (0,0) circle (1cm) circle (0.6cm) circle (0.2cm);

\draw (0,0) circle (1cm);
\draw (0.5,0) circle (0.5cm);
\draw (0,0.5) circle (0.5cm);
\draw (-0.5,0) circle (0.5cm);
\draw (0, -0.5) circle (0.5cm);
```

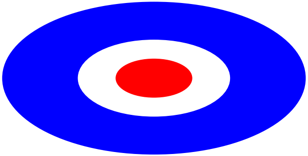
Ellipses



`\draw` (a, b) `ellipse` $(r_1\ dim$ `and` $r_2\ dim)$`;`

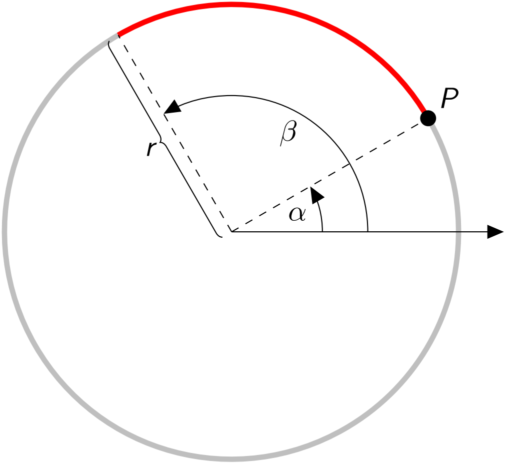
Ellipses: examples





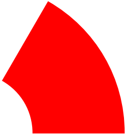
```
\draw (0,0) ellipse (2cm and 1cm);
\draw (0,0) ellipse (0.5cm and 1cm);
\draw (0,0) ellipse (0.5cm and 0.25cm);

\draw[fill,color=blue] (0,0) ellipse (2cm and 1cm);
\draw[fill,color=white] (0,0) ellipse (1cm and 0.5cm);
\draw[fill,color=red] (0,0) ellipse (0.5cm and 0.25cm);
```

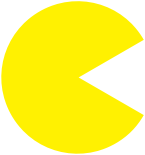
Circular arcs

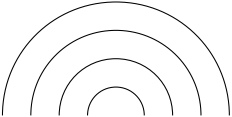


\draw (P) arc ($\alpha : \beta : r$ *dim*);

Circular arcs: examples

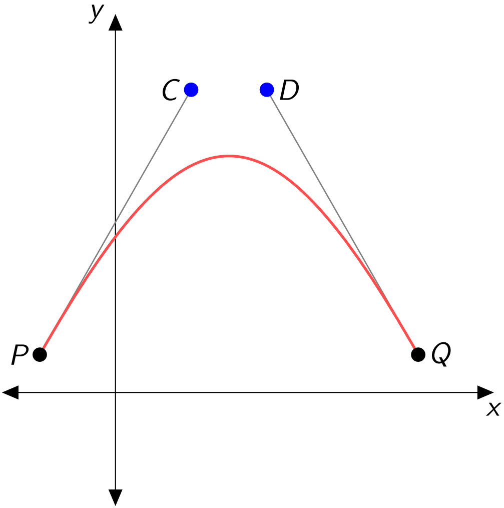






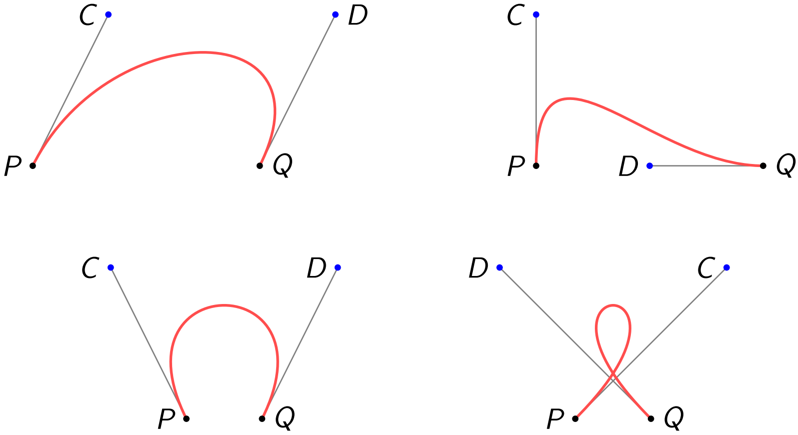
```
\draw[fill, color=red] (0:0.4cm) -- (0:1cm)
                       arc (0:60:1cm) -- (60:0.4cm)
                       arc (60:0:0.4cm) -- cycle;

\draw[fill, color=yellow]
      (0,0) -- (30:1cm) arc (30:330:1cm) -- cycle;

\draw (0.25,0) arc (0:180:0.25cm)
      (0.5,0) arc (0:180:0.5cm)
      (0.75,0) arc (0:180:0.75cm)
      (1,0) arc (0:180:1cm);
```

Bézier curves with named coordinates

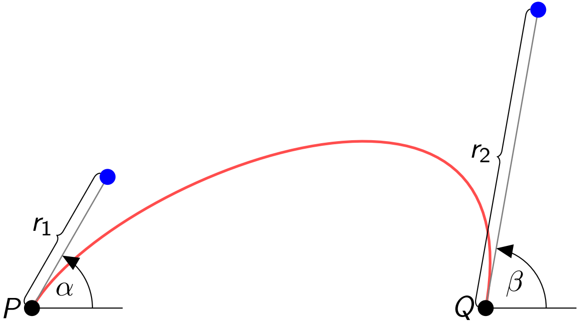


```
\draw (P) .. controls (C) and (D) .. (Q);
```

Bézier curves: examples

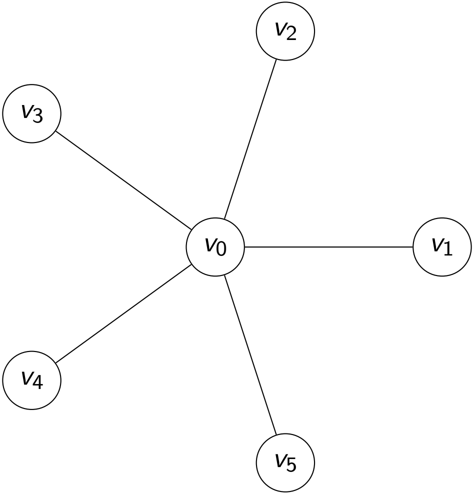


Bézier curves with relative coordinates



\draw (P) .. controls +($\alpha : r_1$ *dim*)
                   and +($\beta : r_2$ *dim*) .. (Q);

Using nodes

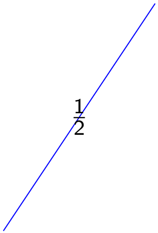


```
\path (0:0cm) node[draw,shape=circle] (v0) {$v_0$};
```

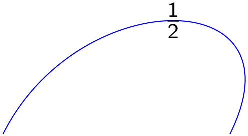
Using nodes: details

```
\begin{tikzpicture}
  \tikzstyle{every node}=[draw,shape=circle];
  \path (0:0cm)    node (v0) {$v_0$};
  \path (0:1cm)    node (v1) {$v_1$};
  \path (72:1cm)   node (v2) {$v_2$};
  \path (2*72:1cm) node (v3) {$v_3$};
  \path (3*72:1cm) node (v4) {$v_4$};
  \path (4*72:1cm) node (v5) {$v_5$};

  \draw (v0) -- (v1)
        (v0) -- (v2)
        (v0) -- (v3)
        (v0) -- (v4)
        (v0) -- (v5);
\end{tikzpicture}
```

Nodes on a line or curve

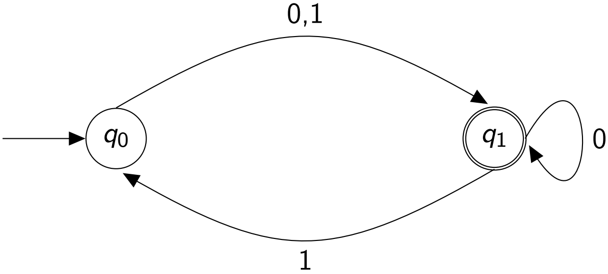


```
\draw[blue] (0,0) -- (2,3) node[black, pos=0.5]{$\frac{1}{2}$};
```



```
\draw[blue] (0,0) .. controls (1,2) and (4,2) .. (3,0)
            node[black, pos=0.5]{$\frac{1}{2}$};
```

Using nodes to draw automata



Using nodes to draw automata: details

```
\begin{tikzpicture}[>=triangle 45]
  % Nodes
  \path (0,0) node[draw,shape=circle] (q0) {$q_0$};
  \path (1,0) node[draw,double,shape=circle] (q1) {$q_1$};

  % Initial state
  \draw[->] (-0.3,0) -- (q0);

  % Edges
  \draw[->, shorten >=1mm]
    (q0.north) ..  controls +(30:0.5cm) and +(150:0.5cm)
               ..  (q1.north) node[pos=0.5,above] {0,1};
  \draw[->, shorten >=1mm]
    (q1.south) ..  controls +(210:0.5cm) and +(330:0.5cm)
               ..  (q0.south) node[pos=0.5,below] {1};
  \draw[->, shorten >=1mm]
    (q1.east) ..  controls +(60:0.4cm) and +(-60:0.4cm)
              ..  (q1.east) node[pos=0.5,right] {0};
\end{tikzpicture}
```

Using nodes to draw automata: details

```
\begin{tikzpicture}[>=triangle 45]
  % Nodes
  \path (0,0) node[draw,shape=circle] (q0) {$q_0$};
  \path (1,0) node[draw,double,shape=circle] (q1) {$q_1$};

  % Initial state
  \draw[->] (-0.3,0) -- (q0);

  % Edges
  \draw[->, shorten >=1mm]
    (q0.north) ..  controls +(30:0.5cm) and +(150:0.5cm)
               ..  (q1.north) node[pos=0.5,above] {0,1};
  \draw[->, shorten >=1mm]
    (q1.south) ..  controls +(210:0.5cm) and +(330:0.5cm)
               ..  (q0.south) node[pos=0.5,below] {1};
  \draw[->, shorten >=1mm]
    (q1.east) ..  controls +(60:0.4cm) and +(-60:0.4cm)
              ..  (q1.east) node[pos=0.5,right] {0};
\end{tikzpicture}
```

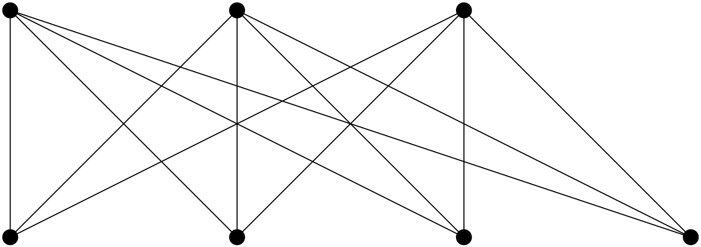
Using nodes to draw automata: details

```
\begin{tikzpicture}[>=triangle 45]
  % Nodes
  \path (0,0) node[draw,shape=circle] (q0) {$q_0$};
  \path (1,0) node[draw,double,shape=circle] (q1) {$q_1$};

  % Initial state
  \draw[->] (-0.3,0) -- (q0);

  % Edges
  \draw[->, shorten >=1mm]
    (q0.north) ..  controls +(30:0.5cm) and +(150:0.5cm)
               ..  (q1.north) node[pos=0.5,above] {0,1};
  \draw[->, shorten >=1mm]
    (q1.south) ..  controls +(210:0.5cm) and +(330:0.5cm)
               ..  (q0.south) node[pos=0.5,below] {1};
  \draw[->, shorten >=1mm]
    (q1.east) ..  controls +(60:0.4cm) and +(-60:0.4cm)
              ..  (q1.east) node[pos=0.5,right] {0};
\end{tikzpicture}
```

Using nodes to draw automata: details

```
\begin{tikzpicture}[>=triangle 45]
  % Nodes
  \path (0,0) node[draw,shape=circle] (q0) {$q_0$};
  \path (1,0) node[draw,double,shape=circle] (q1) {$q_1$};

  % Initial state
  \draw[->] (-0.3,0) -- (q0);

  % Edges
  \draw[->, shorten >=1mm]
    (q0.north) ..  controls +(30:0.5cm) and +(150:0.5cm)
               ..  (q1.north) node[pos=0.5,above] {0,1};
  \draw[->, shorten >=1mm]
    (q1.south) ..  controls +(210:0.5cm) and +(330:0.5cm)
               ..  (q0.south) node[pos=0.5,below] {1};
  \draw[->, shorten >=1mm]
    (q1.east) ..  controls +(60:0.4cm) and +(-60:0.4cm)
              ..  (q1.east) node[pos=0.5,right] {0};
\end{tikzpicture}
```

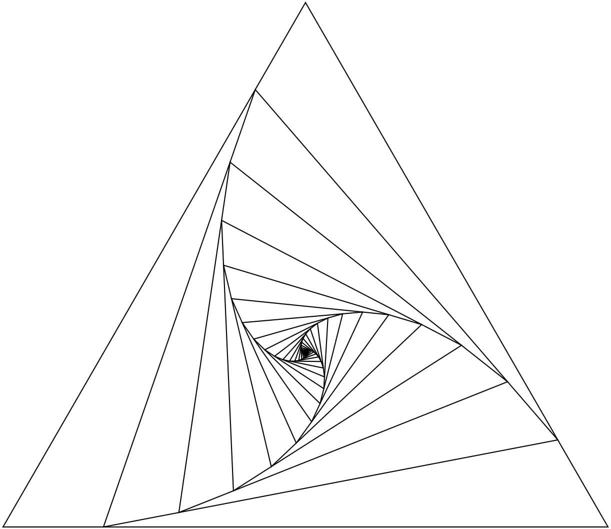
Using loops



Using loops: details

```
\begin{tikzpicture}
  \foreach \i in {1,...,4} {
     \path (\i,0) coordinate (X\i);
     \fill (X\i) circle (1pt);
  }
  \foreach \j in {1,...,3} {
     \path (\j,1) coordinate (Y\j);
     \fill (Y\j) circle (1pt);
  }
  \foreach \i in {1,...,4} {
    \foreach \j in {1,...,3} {
       \draw (X\i) -- (Y\j);
    }
  }
\end{tikzpicture}
```

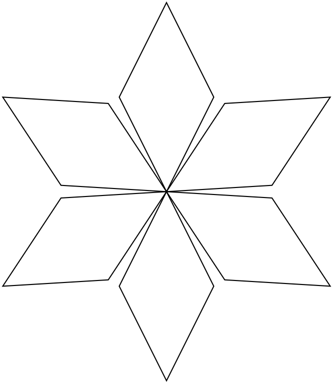
Spiral



Spiral: details

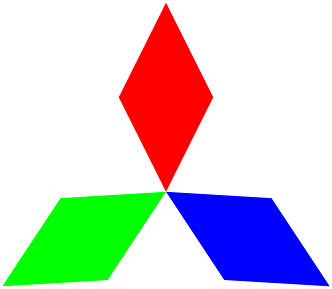
```
% Define the orginal triangle
\path   ( 0 ,  0) node[shape=coordinate](lastA){}
      ++( 60:1cm) node[shape=coordinate](lastB){}
      ++(-60:1cm) node[shape=coordinate](lastC){};

% Draw the orginal triangle
\draw (lastA) -- (lastB) -- (lastC) -- cycle;

\foreach \x in {1,...,60}{ % Draw 60 "sub-triangles"
  % Move A toward C, B towards A, and C towards B
  \path (lastA) -- (lastC) node[shape=coordinate,pos=.166](A){};
  \path (lastB) -- (lastA) node[shape=coordinate,pos=.166](B){};
  \path (lastC) -- (lastB) node[shape=coordinate,pos=.166](C){};

  \draw (A) -- (B) -- (C) -- cycle; % Draw sub-triangle

  \path (A) node[shape=coordinate](lastA){}; % Update positions
  \path (B) node[shape=coordinate](lastB){};
  \path (C) node[shape=coordinate](lastC){};
}
```

Rotation



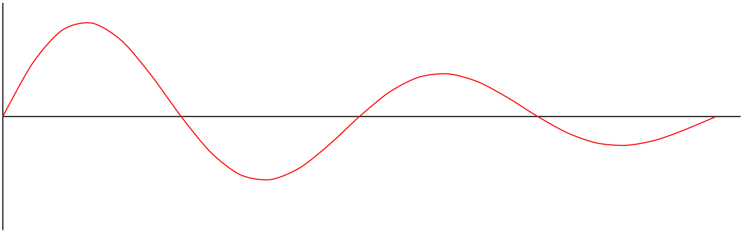
```
\foreach \alpha in {0,60,...,300} {
    \draw[rotate=\alpha] (0,0) -- (0.5,1) --
                         (0,2) -- (-0.5,1) -- cycle;
}
```

Rotation



```
\foreach \alpha / \c in {0/red,120/green,240/blue} {
  \fill[rotate=\alpha,color=\c]
    (0,0) -- (0.5,1) -- (0,2) -- (-0.5,1) -- cycle;
}
```

Function plots



```
\draw[smooth,domain=0:6.28,color=red]
      plot function{sin(2*x)*exp(-x/4)};
```

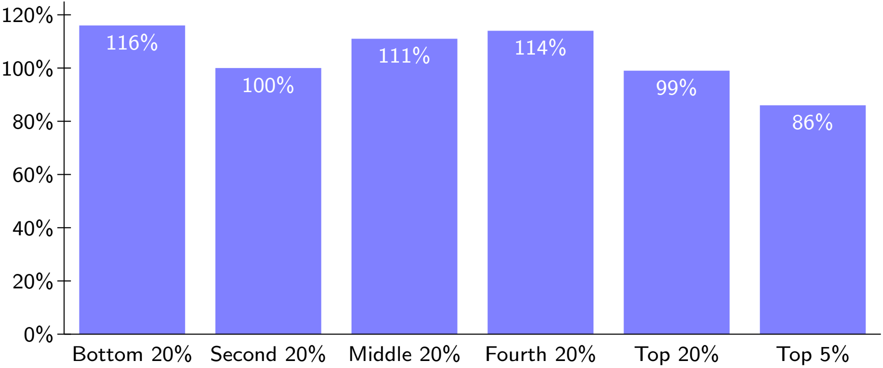
Plot Types

```
\draw plot coordinates {point sequence};
\draw plot file {filename};
\draw plot function {gnuplot formula};
```

Useful options:

- `mark`: places a given mark at each point of the plot.
- `smooth`, `smooth cycle`: points are connected by a (closed) smooth curve.
- `xcomb`, `ycomb`: makes a horizontal or vertical bar diagram.
- `line width`: sets the size of line to use.

Rising Together
Change in Family Income, 1947–1979

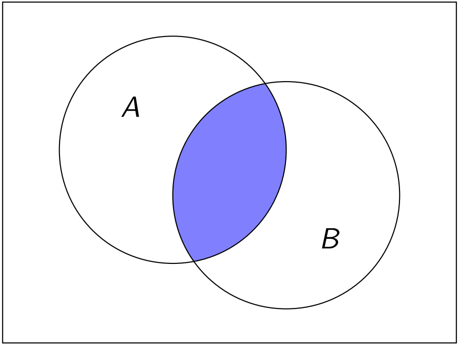


Original graph from `www.faireconomy.org/research/income_charts.html`

Plot: details

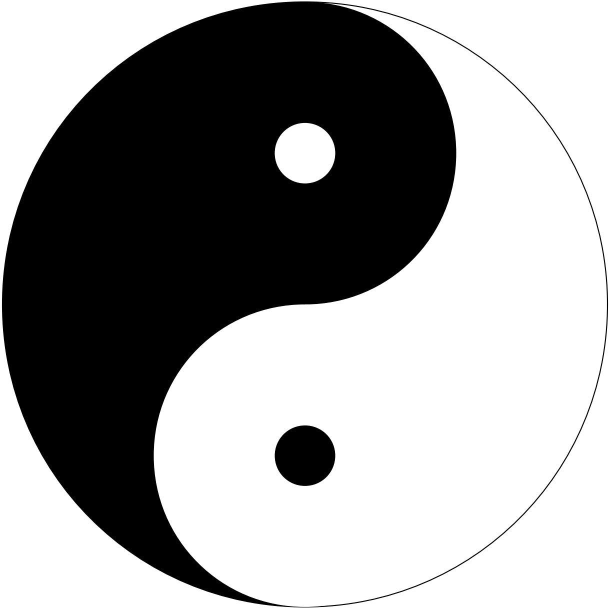
```
\begin{tikzpicture}[x=1.8cm,y=1pt]
  % Bars
  \draw[ycomb, color=blue!50, line width=1.4cm]
    plot coordinates {(0,116) (1,100) (2,111)
                      (3, 114) (4,99) (5,86)};
  % Axes
  \draw (-0.5,0) -- (-0.5,125);
  \draw (-0.5,0) -- (5.5,0);
    ...
\end{tikzpicture}
```

Clipping

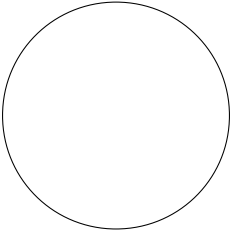


```
\clip (-0.5,0.2) circle (1cm);
\clip (0.5,-0.2) circle (1cm);
\fill[color=blue!50] (-2,1.5) rectangle (2,-1.5);
```

Yin-Yang

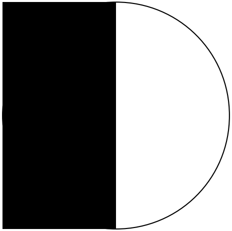


Yin-Yang: details



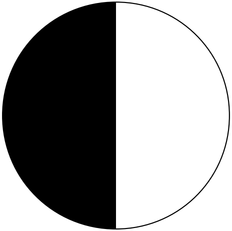
```
\draw (0,0) circle (1cm);
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
\fill[black] (90:0.5cm) circle (0.5cm);
\fill[white] (270:0.5cm) circle (0.5cm);
\fill[white] (90:0.5cm) circle (0.1cm);
\fill[black] (270:0.5cm) circle (0.1cm);
```

Yin-Yang: details



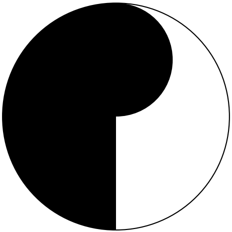
```
\draw (0,0) circle (1cm);
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
\fill[black] (90:0.5cm) circle (0.5cm);
\fill[white] (270:0.5cm) circle (0.5cm);
\fill[white] (90:0.5cm) circle (0.1cm);
\fill[black] (270:0.5cm) circle (0.1cm);
```

Yin-Yang: details



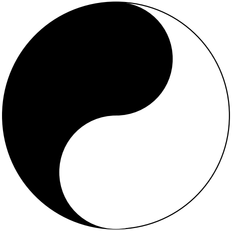
```
\draw (0,0) circle (1cm);
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
\fill[black] (90:0.5cm) circle (0.5cm);
\fill[white] (270:0.5cm) circle (0.5cm);
\fill[white] (90:0.5cm) circle (0.1cm);
\fill[black] (270:0.5cm) circle (0.1cm);
```

Yin-Yang: details



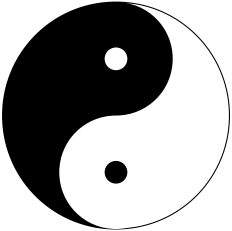
```
\draw (0,0) circle (1cm);
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
\fill[black] (90:0.5cm) circle (0.5cm);
\fill[white] (270:0.5cm) circle (0.5cm);
\fill[white] (90:0.5cm) circle (0.1cm);
\fill[black] (270:0.5cm) circle (0.1cm);
```

Yin-Yang: details



```
\draw (0,0) circle (1cm);
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
\fill[black] (90:0.5cm) circle (0.5cm);
\fill[white] (270:0.5cm) circle (0.5cm);
\fill[white] (90:0.5cm) circle (0.1cm);
\fill[black] (270:0.5cm) circle (0.1cm);
```

Yin-Yang: details



```
\draw (0,0) circle (1cm);
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
\fill[black] (90:0.5cm) circle (0.5cm);
\fill[white] (270:0.5cm) circle (0.5cm);
\fill[white] (90:0.5cm) circle (0.1cm);
\fill[black] (270:0.5cm) circle (0.1cm);
```

Scope

The scope environment limits which paths are subject to a clipping region or applies options to a list of paths.

```
\begin{scope}
  \clip (0,0) circle (1cm);
  \fill[black] (0cm,1cm) rectangle (-1cm,-1cm);
\end{scope}
```

```
\begin{scope}[font=\scriptsize, anchor=base]
  \path (0,-10) node {Bottom 20\%};
  \path (1,-10) node {Second 20\%};
  \path (2,-10) node {Middle 20\%};
  \path (3,-10) node {Fourth 20\%};
  \path (4,-10) node {Top 20\%};
  \path (5,-10) node {Top 5\%};
\end{scope}
```

Summary

- Ti*k*Z is a language for specifying graphics
- Specifying graphics with a language provides exactness
- Familiar graphics-related concepts: points, lines, etc.
- Meshes well with pdfLaTeX and beamer
- Reasonably comfortable learning curve

Acknowledgments

Thanks to Till Tantau for developing Ti*k*Z!

Questions?

```
\begin{tikzpicture}
  \fill[color=blue!10] (-5,-2.5) rectangle (5,2.5);
  \path (0,0) node {{\Huge \color{blue} Questions?}};
\end{tikzpicture}
```

References

- Beamer: `latex-beamer.sourceforge.net`
- PGF/Ti*k*Z: `www.sourceforge.net/projects/pgf/`
- Animate package: `www.ctan.org/tex-archive/macros/latex/contrib/animate/`
- pdfcrop: `www.ctan.org/tex-archive/support/pdfcrop/`