A crash course on using Ti*k*Z

Ben Coté & Jon Lo Kim Lin
cote@math.ucsb.edu
jlokimlin@math.ucsb.edu

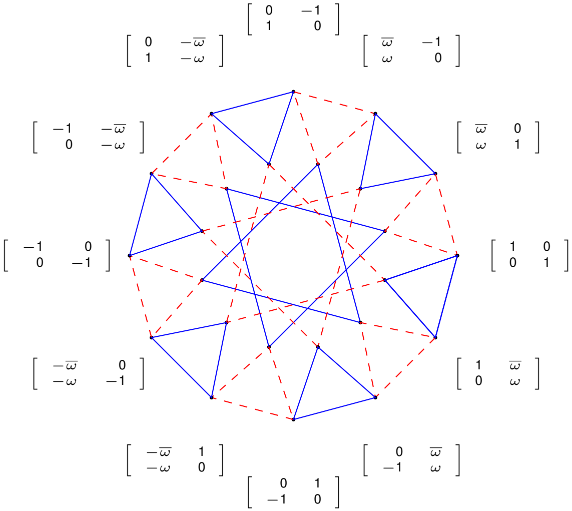
UCSB Hypatian Seminar
04/21/14

Outline

What is Ti*k*Z?

1. A recursive acronym for `TikZ ist kein Zeichenprogramm` (German for "Ti*k*Z is not a drawing program").
2. More specifically, Ti*k*Z is a package for creating pictures.
3. Create anything from rectangles, circles to Koch snowflakes, 3D graphs and animations.
4. Ti*k*Z works very well with Beamer (they were written by the same person!). Muck thanks and gratitude to Till Tantau.

An image by Ben Coté


$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & -\overline{\omega} \\ 1 & -\omega \end{bmatrix}$
$\begin{bmatrix} \overline{\omega} & -1 \\ \omega & 0 \end{bmatrix}$
$\begin{bmatrix} -1 & -\overline{\omega} \\ 0 & -\omega \end{bmatrix}$
$\begin{bmatrix} \overline{\omega} & 0 \\ \omega & 1 \end{bmatrix}$
$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
$\begin{bmatrix} -\overline{\omega} & 0 \\ -\omega & -1 \end{bmatrix}$
$\begin{bmatrix} 1 & \overline{\omega} \\ 0 & \omega \end{bmatrix}$
$\begin{bmatrix} -\overline{\omega} & 1 \\ -\omega & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & \overline{\omega} \\ -1 & \omega \end{bmatrix}$
$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

Ben Coté’s source code

```
\begin{tikzpicture}[scale=1.25]
  \foreach \x in {15,75,...,330} {
    \filldraw[black] (\x:1cm) circle(0.4pt); % dots at each point
    % lines across inside
    \draw[blue] (\x:1cm) -- (\x-120:1cm);
  }
  \foreach \x in {45,105,...,345} {
    \filldraw[black] (\x:1cm) circle(0.4pt); % dots at each point
    % lines across outside
    \draw[red,dashed] (\x:1cm) -- (\x-120:1cm);
  }
  \foreach \x in {30,90,...,330} {
    \filldraw[black] (\x:1.73205cm) circle(0.4pt); % dots at each point
    % lines across inside
    \draw[red,dashed] (\x:1.73205cm) -- (\x-30:1.73205cm) -- (\x-15:1cm) -- cycle;
  }
  \foreach \x in  {0,60,...,360} {
    \filldraw[black] (\x:1.73205cm) circle(0.4pt); % dots at each point
    % lines across inside
    \draw[blue] (\x:1.73205cm) -- (\x-30:1.73205cm) -- (\x-15:1cm) -- cycle;
  }
  \foreach \x/\a/\b/\c/\d in {
    0/1/0/0/1,
    30/\ow/0/\w/1,
    60/\ow/-1/\w/0,
    90/0/-1/1/0,
    120/0/-\ow/1/-\w,
    150/-1/-\ow/0/-\w,
    180/-1/0/0/-1,
    210/-\ow/0/-\w/-1,
    240/-\ow/1/-\w/0,
    270/0/1/-1/0,
    300/0/\ow/-1/\w,
    330/1/\ow/0/\w
  }
  \draw (\x:2.5cm) node { $\left[ \begin{array}{rr} \a & \b \\  \c & \d \end{array}\right]$};
\end{tikzpicture}
```

Preliminaries

1. You need to add

```
\usepackage{tikz}
```

   to your document preamble. Other commands to put in preamble will be discussed later.

2. When creating a picture use the `tikzpicture` environment. E.g.

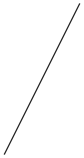
```
\begin{tikzpicture}
  %
  %... code
  %
\end{tikzpicture}
```

Simples straight lines

To draw a line you do something like

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

and you get

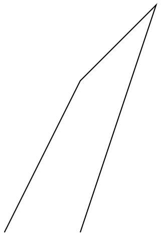


Ti*k*Z automatically draws a line between the points $(0,0)$ and $(1,2)$ and sets up the right space for the figure (by default, coordinates are in centimeters).

You can do a sequence of segments which goes from point to point:

```
\begin{tikzpicture}
  \draw (0,0) --(1,2) -- (2,3) -- (1,0);
\end{tikzpicture}
```

to get



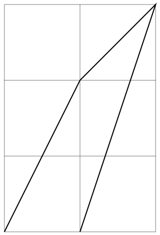
We now have added the grid lines on the graphic to make it clearer. This is done through the command

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

which draws a grid from $(0,0)$ to $(2,3)$.

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

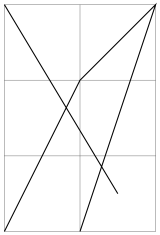
to get



Of course, you can put several lines on the same graph:

```
\begin{tikzpicture}
  \draw [help lines] (0,0) grid (2,3);
  \draw (0,0) --(1,2) -- (2,3) -- (1,0);
  \draw (0,3) -- (1.5,0.5);
\end{tikzpicture}
```

yields



Notice the semi-colons “; ”at the end of lines – they mark the end of instructions. That is, in the last picture we could have written

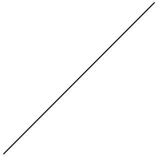
```
      \draw (0,0) --(1,2) -- (2,3) -- (1,0); \draw (0,3) -- (1.5,0.5);
```

without changing the output. You can also add and suppress spaces, for instance in order to make the code easier to read, without changing anything in the output.

Scaling pictures

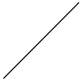
One very useful feature of Ti*k*Z is that you can blow up the picture, by adding an option “scale” to the environment.

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



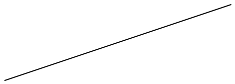
which you can compare to the following:

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



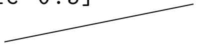
You can scale only one dimension:

```
\begin{tikzpicture}[xscale=3]
  \draw (0,0) -- (1,1);
\end{tikzpicture}
```



or both dimensions in different proportions:

```
\begin{tikzpicture}[xscale=2.5,yscale=0.5]
  \draw (0,0) -- (1,1);
\end{tikzpicture}
```

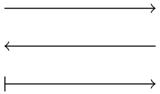


Arrows and the like

You can “decorate” the lines. For instance we can put arrows or bars on one of both extremities:

```
\begin{tikzpicture}
  \draw [->] (0,0) -- (2,0);
  \draw [<-] (0, -0.5) -- (2,-0.5);
  \draw [|->] (0,-1) -- (2,-1);
\end{tikzpicture}
```

which yields



When you draw several segments, the arrows are placed at the extremities of the first and the last segments. This is convenient, among other things to draw axes (we will see later how to label them):

```
\begin{tikzpicture}
  \draw [<->] (0,2) -- (0,0) -- (3,0);
\end{tikzpicture}
```

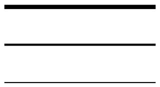
which gives you

Changing the thickness of lines

Other decorations include changing the thickness:

```
\begin{tikzpicture}
  \draw [ultra thick] (0,1) -- (2,1);
  \draw [thick] (0,0.5) -- (2,0.5);
  \draw [thin] (0,0) -- (2,0);
\end{tikzpicture}
```

which gives you

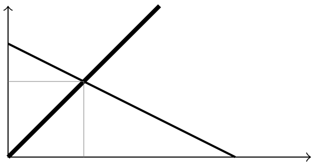


The help lines option

There is also the `help lines` option, discussed earlier, which is made specially to be fine gray lines for showing special points:

```
\begin{tikzpicture}
  \draw [<->] (0,2) -- (0,0) -- (4,0);
  \draw [thick] (0,1.5) -- (3,0);
  \draw [ultra thick] (0,0) -- (2,2);
  \draw [help lines] (1,0) -- (1,1) -- (0,1);
\end{tikzpicture}
```

which yields.

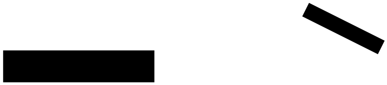


Custom line widths

You can also use custom widths:

```
\begin{tikzpicture}
  \draw [line width=12] (0,0) -- (2,0);
  \draw [line width=0.2cm] (4,.75) -- (5,.25);
\end{tikzpicture}
```

which gives a line 12pt wide (the default dimension for width is point) and another one .2cm wide:

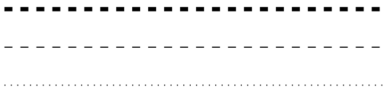


Dashes and dots

You can also make dashed and dotted lines

```
\begin{tikzpicture}
  \draw [dashed, ultra thick] (0,1) -- (5,1);
  \draw [dashed] (0, 0.5) -- (5,0.5);
  \draw [dotted] (0,0) -- (5,0);
\end{tikzpicture}
```

This gives:



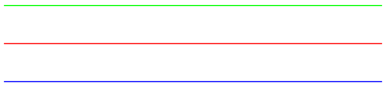
The top line shows you that you can mix types of decorations. You have lots of control over the style of your dotted and dashed lines (see the manual).

Colored lines

And finally, you can color your lines.

```
\begin{tikzpicture}
  \draw [green] (0,1) -- (5,1);
  \draw [red] (0, 0.5) -- (5,0.5);
  \draw [blue] (0,0) -- (5,0);
\end{tikzpicture}
```

We obtain



Taste the rainbow

You have direct access to the following pre-defined colors: red 🟥 , green 🟩 , blue 🟦 , cyan 🟦 , magenta 🟪 , yellow 🟨 , black ⬛ , darkgray[1] ⬛ , light gray ⬜ , brown 🟫 , lime 🟩 , olive 🟫 , orange 🟧 , pink ⬜ , purple 🟥 , teal 🟦 , violet 🟪 , and white ⬜ . And you can define all the colors you might want using the RGB color model (see the manual).

[1] An affront to the King's language

Pictures in the middle of text

By the way, you may wonder how we included these rectangles in the text. Ti*k*Z makes a picture wherever you want; we just typed

```
\begin{tikzpicture}
  \draw [blue, line width=6] (0,0) -- (.5,0);
\end{tikzpicture}
```

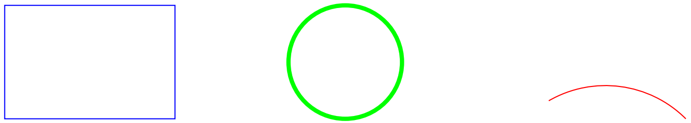
wherever we want in the text. To make these constructions easier to type you can use the command **tikz** (see the manual).

Curves

We are not limited to straight lines:

```
\begin{tikzpicture}[scale=1.5]
  \draw [blue] (0,0) rectangle (1.5,1);
  \draw [green, ultra thick] (3,0.5) circle [radius=0.5];
  \draw [red] (6,0) arc [radius=1, start angle=45, end angle= 120];
\end{tikzpicture}
```

This gives:



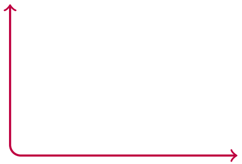
The arc is of radius 1, starts at the point $(6,0)$ leaving it at an angle of 45° and stops when its slope is 120°.

Smoother curves

We can make paths take smoother turns:

```
\begin{tikzpicture}
  \draw [<->, rounded corners, thick, purple] (0,2) -- (0,0) -- (3,0);
\end{tikzpicture}
```

which gives us

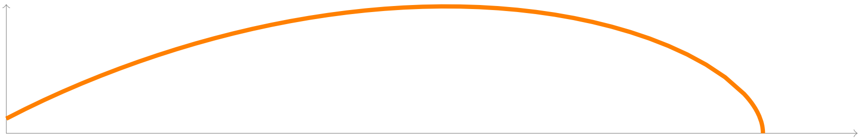


Hard coding graphs

If you want a precise curve you can do it by computing lots of points in a program such as C/C++, Fortran, Python, MATLAB, etc, and then putting them into Ti*kZ*:

```
\begin{tikzpicture}[xscale=25,yscale=5]
  \draw [<->, help lines] (0.6,1.34) -- (0.6,1) -- (1.05,1);
  \draw [orange, ultra thick] (0.6, 1.0385) --
(0.61, 1.06372) -- (0.62, 1.08756) -- (0.63, 1.11012) -- (0.64,
1.13147) -- (0.65, 1.15166) -- (0.66, 1.17074) -- (0.67, 1.18874) -- (0.68,
1.20568) -- (0.69, 1.22157) -- (0.7, 1.23643) -- (0.71, 1.25026) -- (0.72,
1.26307) -- (0.73, 1.27486) -- (0.74, 1.28561) -- (0.75, 1.29534) -- (0.76,
1.30402) -- (0.77, 1.31165) -- (0.78, 1.31821) -- (0.79, 1.32369) -- (0.8,
1.32806) -- (0.81, 1.33131) -- (0.82, 1.3334) -- (0.83, 1.33431) -- (0.84,
1.334) -- (0.85, 1.33244) -- (0.86, 1.32956) -- (0.87, 1.32533) -- (0.88,
1.31966) -- (0.89, 1.3125) -- (0.9, 1.30373) -- (0.91, 1.29325) -- (0.92,
1.2809) -- (0.93, 1.26649) -- (0.94, 1.24976) -- (0.95, 1.23032) -- (0.96,
1.2076) -- (0.97, 1.18065) -- (0.98, 1.14763) -- (0.99, 1.1038) -- (0.991,
1.09836) -- (0.992, 1.09261) -- (0.993, 1.0865) -- (0.994, 1.07994) -- (0.995,
1.07282) -- (0.996, 1.06497) -- (0.997, 1.0561) -- (0.998, 1.04563) -- (0.999,
1.03209) -- (0.9991, 1.03042) -- (0.9992, 1.02866) -- (0.9993,
1.02679) -- (0.9994, 1.02478) -- (0.9995, 1.0226) -- (0.9996, 1.02019) -- (0.9997,
1.01747) -- (0.9998, 1.01424) -- (0.9999, 1.01005) -- (0.9999,
1.01005) -- (0.99991, 1.00953) -- (0.99992, 1.00898) -- (0.99993,
1.0084) -- (0.99994, 1.00778) -- (0.99995, 1.0071) -- (0.99996,
1.00634) -- (0.99997, 1.00549) -- (0.99998, 1.00448) -- (0.99999, 1.00317) -- (1,1);
\end{tikzpicture}
```

Hard coded graph results

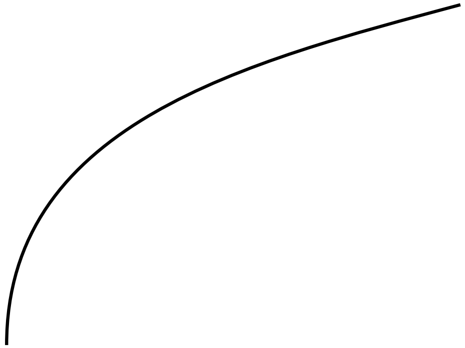


1. This was overkill: We do not need so many points;
2. This can also serve as a reminder that one Ti*k*Z instruction can be spread over several lines and cut arbitrarily over several lines. The marker is the semi-colon, not the end of line!

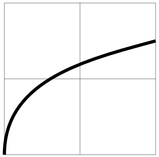
There are a number of ways by which you can do curves without plotting all the points. Here is an easy one:

```
\begin{tikzpicture}[scale=3]
  \draw [very thick] (0,0) to [out=90,in=195] (2,1.5);
\end{tikzpicture}
```

This gives us a curve from $(0, 0)$ to $(2, 1.5)$ which leaves at an angle of 90° and arrive at an angle of 195°: Notice that we replaced the `--` with `to`.



Replacing `--` with `to`



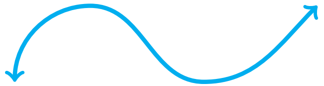
1. When the curves goes out of $(0,0)$, you put a needle with one extremity on the starting point and the other one facing right and you turn it counterclockwise until it is tangent to the curve. The angle by which you have to turn the needle gives you the out angle.
2. When the curves goes in at $(2,1.5)$, you put a needle with one extremity on the arrival point and the other one facing right and you turn it counterclockwise until it is tangent to the curve. The angle by which you have to turn the needle gives you the in angle.

Several `to` instructions

As with straight lines you can put several `to` instructions in the same Ti*k*Z instruction:

```
\begin{tikzpicture}
  \draw [<->,ultra thick, cyan]
  (0,0) to [out=90,in=180]
  (1,1) to [out=0,in=180]
  (2.5,0) to [out=0,in=-135] (4,1);
\end{tikzpicture}
```

we obtain:

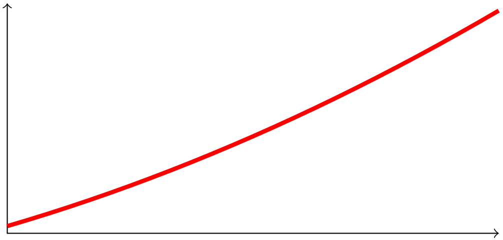


Plotting user-defined functions

Ti*k*Z also has a math engine which enables you to plot functions:

```
\begin{tikzpicture}[xscale=13,yscale=3.8]
  \draw [<->] (0,0.8) -- (0,0) -- (0.5,0);
  \draw[red, ultra thick, domain=0:0.5]
  plot (\x, {0.025+\x+\x*\x});
\end{tikzpicture}
```

gives you



The domain instruction shows the range of x which is plotted. In this case we are plotting the function $0.025 + x + x^2$. Note the braces around the function that we plot in

```
    plot (\x, {function});
```

Built-in math functions

- Many mathematical functions are possible; you will probably have enough with

  ```
  factorial(\x), sqrt(\x), exp(\x), ln(\x)
  log2(\x), log10(\x), abs(\x)
  ```

-
  ```
  pow(\x,y), mod(\x, y)
  ```

  which gives x^y and x modulo y.

-
  ```
  round(\x), floor(\x), ceil(\x)
  ```

  rounds x to the nearest integer, the largest integer smaller than x, the smallest integer larger than x.

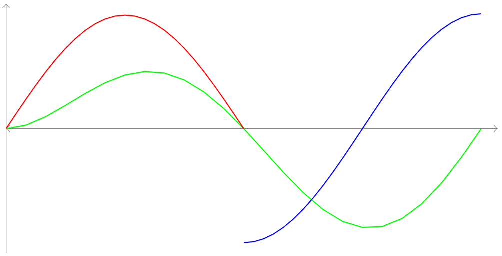
Built-in math functions (continued)

-
  ```
  sin(\x)
  ```
  it assumes that *x* is in degrees; if *x* is expressed in radians use
  ```
  sin(\x r)
  ```
  In a similar fashion,
  ```
  cos(\x), cos(\x r), tan(\x), tan(\x r)
  ```
  We also have
  ```
  min(\x,y), max(\x,y).
  ```
- In mathematical expressions the two following **rational** variables can be useful:
  ```
  e
  ```
  which is equal to 2.718281828, and
  ```
  pi
  ```
  which is equal to 3.141592654.

You can mix functions to compute more complicated expressions (see above for the reason for the `r` parameter in the argument of sin and cos and note the use of `pi` to define the domains):

```
\begin{tikzpicture}[yscale=1.5]
  \draw [help lines, <->] (0,0) -- (6.5,0);
  \draw [help lines, ->] (0,-1.1) -- (0,1.1);
  \draw [green,domain=0:2*pi]
  plot (\x, {(sin(\x r)* ln(\x+1))/2});
  \draw [red,domain=0:pi]
  plot (\x, {sin(\x r)});
  \draw [blue, domain=pi:2*pi]
  plot (\x, {cos(\x r)*exp(\x/exp(2*pi))});
\end{tikzpicture}
```

which gives you

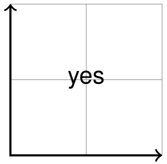


Putting labels in pictures

When you construct a picture, in 99% of cases you also need to put labels. This is easy! Let us start by seeing how we would place some text in a picture.

```
\begin{tikzpicture}
  \draw [help lines] (0,0) grid (2,2);
  \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
  \node at (1,1) {yes};
\end{tikzpicture}
```

yields



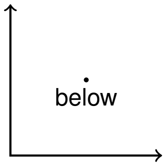
Notice how the "yes" is positioned: the center of its "baseline" is at $(1,1)$.

Relative positioning

Sometimes you want a label to be situated relative to a point. TikZ has neat commands for this. For instance you can write

```
\begin{tikzpicture}
  \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
  \draw[fill] (1,1) circle [radius=0.025];
  \node [below] at (1,1) {below};
\end{tikzpicture}
```

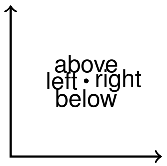
to get



You are not limited to put things below a point:

```
\begin{tikzpicture}
  \draw [thick, <->] (0,2) -- (0,0) -- (2,0);
  \draw [fill] (1,1) circle [radius=0.025];
  \node [below] at (1,1) {below};
  \node [above] at (1,1) {above};
  \node [left] at (1,1) {left};
  \node [right] at (1,1) {right};
\end{tikzpicture}
```

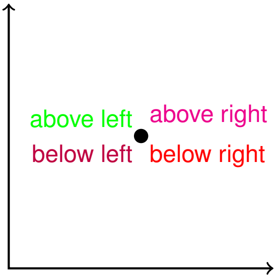
yields



And, you can also mix and match

```
\begin{tikzpicture}[scale=3.5]
  \draw [thick, <->] (0,1) -- (0,0) -- (1,0);
  \draw[fill] (.5,.5) circle [radius=0.025];
  \node [below right, red] at (.5,.5) {below right};
  \node [above left, green] at (.5,.5) {above left};
  \node [below left, purple] at (.5,.5) {below left};
  \node [above right, magenta] at (.5,.5) {above right};
\end{tikzpicture}
```

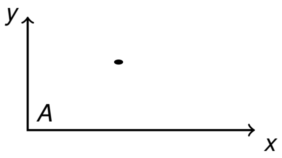
yields



Labeling axes and points

```
\begin{tikzpicture}[xscale=3, yscale=1.5]
  \draw [thick, <->] (0,1) -- (0,0) -- (1,0);
  \node [below right] at (1,0) {$x$};
  \node [left] at (0,1) {$y$};
  \draw [fill] (.4,.6) circle [radius=.5pt];
  \node[above right] (.4,.6) {$A$};
\end{tikzpicture}
```

gives us



You can avoid some typing by mixing nodes in the middle of paths. For instance the last figure could have been written as follows:

```
\begin{tikzpicture}[xscale=3, yscale]
  \draw [thick, <->] (0,1) node [left] {$y$}
  -- (0,0) -- (1,0) node [below right] {$x$};
  \draw[fill] (.4,.6) circle [radius=.5pt]
  node [above right] (.4,.6) {$A$};
\end{tikzpicture}
```

which would have given exactly the same result. Note that the node is put after the point to which it is attached and that we suppress the \ in

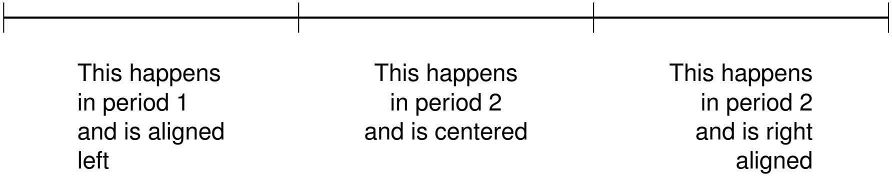
```
    \node
```

Fancy nodes

You may want to put several lines in your "node" (this is convenient when drawing time lines for instance). This can be done by using the standard LaTeX for indicating a new line but you must tell Ti*k*Z how to align things. From

```
\begin{tikzpicture}[xscale=1.3]
  \draw [thick] (0,0) -- (9,0);
  \draw (0,-.2) -- (0, .2);
  \draw (3,-.2) -- (3, .2);
  \draw (6,-.2) -- (6, .2);
  \draw (9,-.2) -- (9, .2);
  \node[align=left, below] at (1.5,-.5)%
       {This happens\\in period 1\\and is aligned\\ left};
  \node[align=center, below] at (4.5,-.5)%
       {This happens\\in period 2\\and is centered};
  \node[align=right, below] at (7.5,-.5)%
       {This happens\\in period 2\\and is right\\aligned};
\end{tikzpicture}
```

we obtain



Ti*k*Z libraries

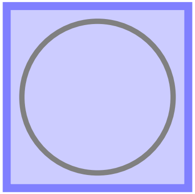
- To provide simpler methods to create complex structures libraries for Ti*k*Z has `\usetikzlibary`
- Different examples are: `decorations.fractals`, `arrows`, `backgrounds`, `intersections`, and `shapes.geometric`.

And now for some examples

First you must add

```
    \usetikzlibrary{backgrounds}
```

to your preamble

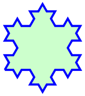


```
\begin{tikzpicture}
  [background rectangle/.style={fill=blue!20,
      draw=blue!50,line width=3pt},
    show background rectangle]
  \draw[line width=2pt,color=gray]
  (2,2) circle[radius=1];
\end{tikzpicture}
```

- For the Koch snowflake, first you must add

```
\usetikzlibrary{decorations.fractals}
```

  to your preamble



```
\begin{tikzpicture}[decoration=Koch
    snowflake,draw=blue,fill=green!20,thick]
  \filldraw decorate{ decorate{ (0,0) --
      ++(60:1) -- ++(-60:1) -- cycle}};
\end{tikzpicture}
```

- Don’t forget to put the correct Ti*k*Z library in your preamble.

Conclusion

1. Ti*k*Z is an enormous program. You will find commands to draw hierarchical trees, to draw lots of different types of shapes, to do some elementary programming, to align elements of a picture in a matrix frame, to decorate nodes, to compute the intersections of paths, etc.
2. If you conundrum is not addressed in these slides it's probably somewhere in the exhaustive manual available at tikzpgfmanual
3. Post questions on TeX - LaTeX Stack Exchange, a question and answer site for users of TeX, LaTeX, ConTeXt, and related typesetting systems. It's 100% free, no registration required.

Thank you for your time