Math 291: Lecture 9

Dr. Fagerstrom

Minnesota State University Moorhead
web.mnstate.edu/fagerstrom
fagerstrom@mnstate.edu

April 11, 2019

Outline

Outline

Inputting and Including Other Files

- Sometimes, when we are creating a very large documents, you may want to create the final document by piecing several smaller documents together.
- LaTeX has nice commands for doing this:
- The \input{*filename*} command
  - The contents of *filename.tex* are read by the compiler as if they are actually typed into your code at the place where the input command is.
  - \input can be used in the preamble (in fact, it can be used to **create** the preamble).
  - \input commands can be nested (in examplefile.tex, the input command is used to input file1, which inputs file2, which inputs file3, etc., all of which ends up in examplefile.tex). Limit is computer capability, not anything specific to LaTeX.

Inputting and Including Other Files

- There are no restrictions to what can be inputted using the input command.
  - So your main document can literally consist of:
    \documentclass{*class*}
    \input{file1}
    \begin{document}
    \input{file2}
    \input{file3}
    \input{file4}
    \input{file5}
    \end{document}

Inputting and Including Other Files

- Make sure you first create a separate .tex file that contains all of the material that we want to include as a portion of the larger document
- That file has *only* the material we want to include (no preamble or packages or begin and end document commands unless they are not included in the main file)
- Using \input does have the compiler re-create the entire document each time the main document is built.
- If you use \include instead you have the option of only compiling the bit that you have just edited.
- But, \include has a lot of restrictions on it (can't be in the preamble, can't be nested, can't include new counters, etc.)
- For more info on how to use \include, look it up!

Outline

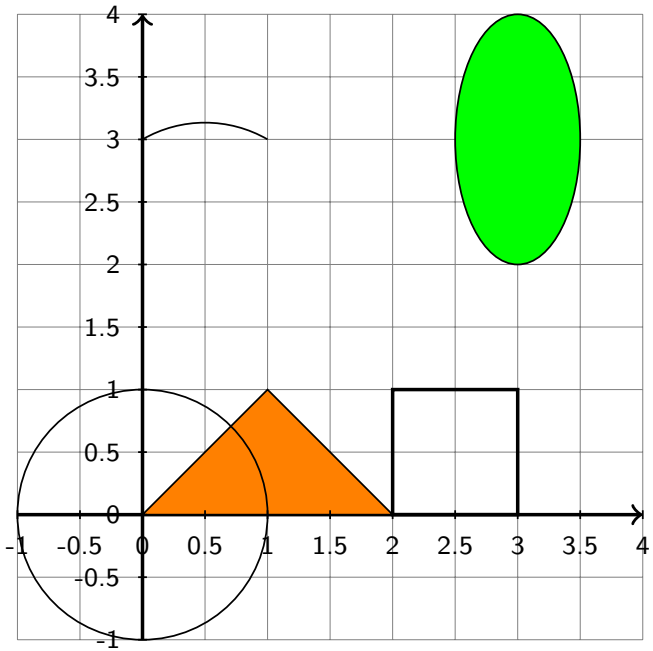
What is TikZ?

- Ti*k*Z stands for "TikZ ist kein Zeichenprogramm", which translates to "Ti*k*Z is not a drawing program".
- But Ti*k*Z IS a drawing program with a lot of bells and whistles.
- This lecture is based on some of the examples from the TikZ manual.
- Here is a link to a nice brief introduction to TikZ.
- Note: Ti*k*Z is especially useful for drawing graphs and diagrams. (It's not very good at graphing functions.)

Outline

Example 1 Result

I'm going to create the following picture, working up to it in bits and pieces.



The tikZ Environment

- Use the package tikz.
- In the preamble, add the command:

```
\usetikzlibrary{calc,intersections,through,backgrounds}
```

- The environment's name is tikzpicture (so use \begin{tikzpicture} and the appropriate end statement).
- Within that environment are put the ti*k*Z commands.
- All ti*k*Z commands end with a semicolon;

Basic Commands

- Some basic commands are:
  - \draw (x0,y0) -- (x1,y1) -- (x2,y2);
  - \filldraw (x0,y0) -- (x1,y1) -- (x2,y2);
  - \draw (x0,y0) -- (x1,y1) -- (x2,y2) -- cycle;
  - \draw (h,k) circle (r);
  - \draw (h,k) ellipse (m and M);
  - \draw (x0,y0) arc (a:b:r);
  - \draw (s,w) rectangle (n,e);
  - \draw (s,w) grid (n,e);

Size and scaling

- tikZ sets aside the amount of space needed automatically, so you don't have to pre-determine the size of your graphic object
- The default unit is 1 cm.
- You can scale your entire picture in a couple of ways:
  - \begin{tikzpicture}[scale=0.2]
    makes the entire picture 20% of the original size (any decimal value can be used for the scale, including values larger than one if you want to scale up instead of scale down)
  - \begin{tikzpicture}[xscale=2.5,yscale=0.5]
    scales in the x- and y-directions separately, and you are allowed to do only one or the other if you desire.
  - \begin{tikzpicture}[x=1in,y=10.5pt]
    sets the size of a unit in the x-direction to be 1 inch, and a unit in the y-direction to be 10.5 points.
    any standard unit of measurement can be used (cm, mm, in, pt, em, etc.)

Basic Commands: draw and filldraw

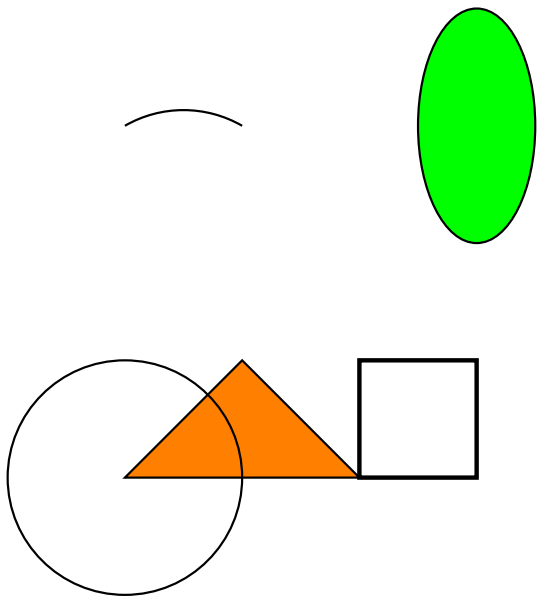
- The draw and filldraw commands that just use a bunch of points
  - draws a connected stream of line segments with the given points used as the endpoints of the line segments
  - adding 'cycle' at the end connects the start/stop points to create a polygon
  - Use (0in,2in) instead of (0,2) if you want to use a different unit than the current default unit
    If you used a unit to set your default, it will end up at a y of 2in.
    If you used a scale command to set your default, then that scale still applies to these, so it might not end up at 2in.
    Using scale is useful if you want to maintain the aspect ratio, though!
  - change 'draw' to 'filldraw' to fill in your object
    If you did not use cycle, it fills the object as if you had connected the start and stop points, but note that there will not be an edge drawn on that side of the polygon.

Basic Commands: drawing shapes and arcs

- Use draw with the cycle command for polygons like triangles
- \draw (h,k) circle (r);
  circle with center (h,k) and radius r
- \draw (h,k) ellipse (m and M);
  ellipse with center (h,k), m and M are half the length of the axes in the x- and y-directions, respectively
- \draw (x0,y0) arc (a:b:r);
  an arc where (x0,y0) is the starting point, a is the starting angle, b is the ending angle and r is the radius
- \draw (s,w) rectangle (n,e);
  rectangle where (s,w) is the southwest corner (n,e) is the northeast corner

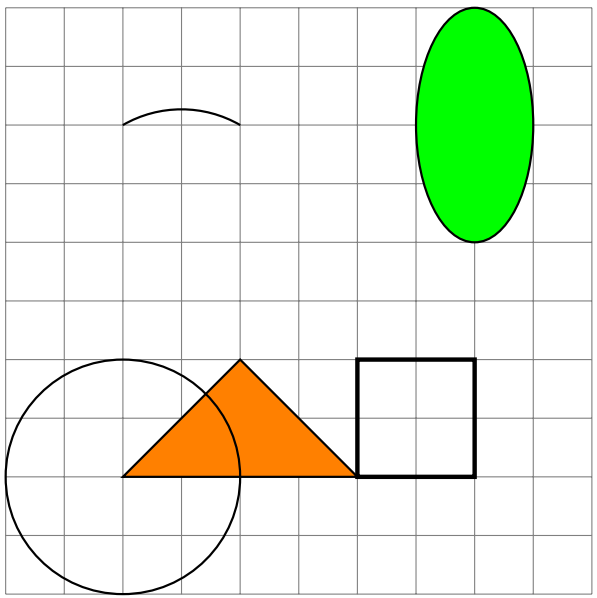
Starting the Example

```
\begin{tikzpicture}
\draw[fill=orange, draw=black] (0,0) -- (1,1) -- (2,0) -- cycle;
\draw (0,0) circle (1);
\draw[fill=green] (3,3) ellipse (0.5 and 1);
\draw [thick] (2,0) rectangle (3,1);
\draw (1,3) arc (60:120:1);
\end{tikzpicture}
```



Grid Construction

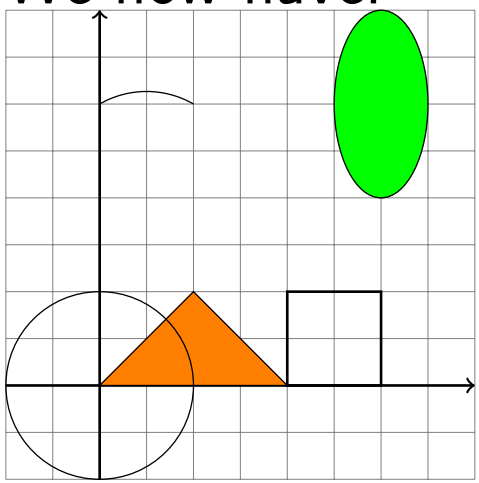
- To get the background grid: \draw[options] (s,w) grid (n,e);
  Note: Add it *first*, so it is 'below' the other objects.
- We'll use the command: \draw[step=.5] (-1,-1) grid (4,4);
- The default lines on the grid are so dark and thick that they are distracting. We'll add the option help lines:
  \draw[step=.5,help lines] (-1,-1) grid (4,4);



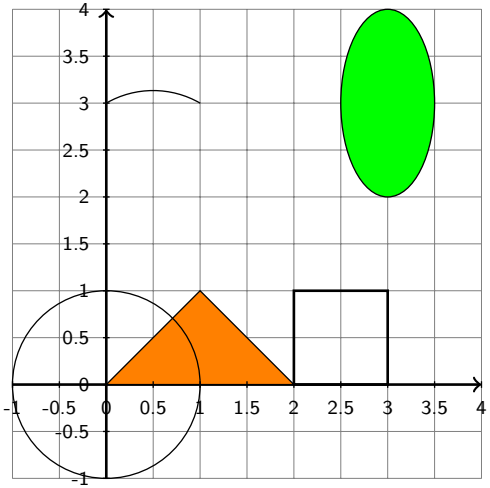
Axes and Arrows

- We still need the axes (separate from the background grid).
- They are really just lines (so 'draw' command) with arrowheads.
  \draw[-⟩,thick] (-1,0) – (4,0);
  \draw[-⟩,thick] (0,-1) – (0,4);
- Arrow options include ⟨-⟩ (both ends) and ⟨- (only the first end), and you can reverse them in you want them backwards (point into the line instead of out of it)

We now have:



And we want:



All that is missing are the scales on the axes.

Adding Text

- In order to add text we need the next basic command in Ti*k*Z: node.
- The basic command is:
  \node at (x0,y0) {what the node says}
- But you can also add a node command within a draw command when the node is connected to what is being drawn.
- Doing it within the draw command for the tic mark, we can add:
  \draw (0.5,1pt) – (0.5,-1pt) node[anchor=north] {\small 0.5};
- Note that the direction gives the direction of the anchor point from the node/label (not the direction of the node/label from the anchor point).
- Adding all labels one at a time seems like a lot of work. Ti*k*Z makes this easier.

For-Loops

- TikZ can use loops, as in a computer program.
- The syntax for for loops in tikZ is as follows.

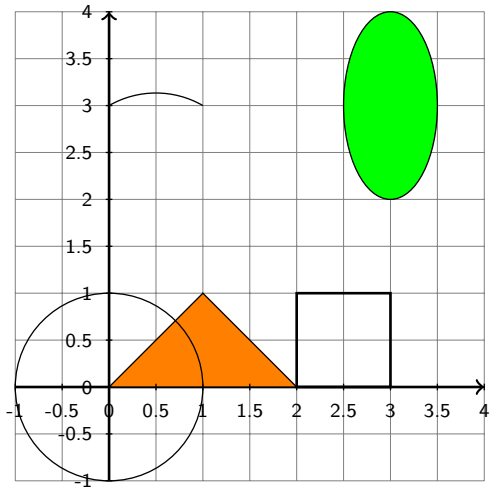
```
\foreach \x in {values x can take on}
whatever you want to draw;
```

- Note that the variable is essentially \x, not just x
- To draw the rest of our axis labels we would add the commands

```
\foreach \x in {-1,-0.5,...,4} \draw (\x,1pt) -- (\x,-1pt) node[anchor=north]{\small \x};
\foreach \y in {-1,-0.5,...,4} \draw (1pt,\y) -- (-1pt,\y) node[anchor=east]{\tiny \y};
```

Finishing the example

Output:



Code:

```
\begin{tikzpicture}
\draw[step=.5,help lines] (-1,-1) grid (4,4);
\draw[->,thick] (-1,0) -- (4,0);
\draw[->,thick](0,-1) -- (0,4);
\draw[fill=orange, draw=black] (0,0) -- (1,1) -- (2,0) -- cycle;
\draw (0,0) circle (1);
\draw[fill=green] (3,3) ellipse (0.5 and 1);
\draw [thick] (2,0) rectangle (3,1);
\draw (1,3) arc (60:120:1);
\foreach \x in {-1,-0.5,...,4} \draw (\x,1pt) -- (\x,-1pt) node[anchor=north]{\tiny \x};
\foreach \y in {-1,-0.5,...,4} \draw (1pt,\y) -- (-1pt,\y) node[anchor=east]{\tiny \y};
\end{tikzpicture}
```

Reference: TikZ Options

- To shade an object we can use the filldraw command.
- To get an orange triangle with a black outline:
  \filldraw[fill=orange,draw=black] (0,0) -- (1,1) -- (2,0) -- cycle;
- Note that the default fillcolor and drawcolor are black, and will be used unless you specify otherwise.
- The default available colors are:
  red, green, blue, cyan, magenta, yellow, black, gray, darkgray, lightgray, brown, lime, olive, orange, pink, purple, teal, violet, white
- To get a color gradient, use the `\shadedraw` command.
- To get a color gradient in a triangle:
  `\shadedraw[left color=white, right color=green, draw=black] (0,0)--(1,1)--(2,0)--cycle;`
- A similar command with a bottom color and a top color can also be used.

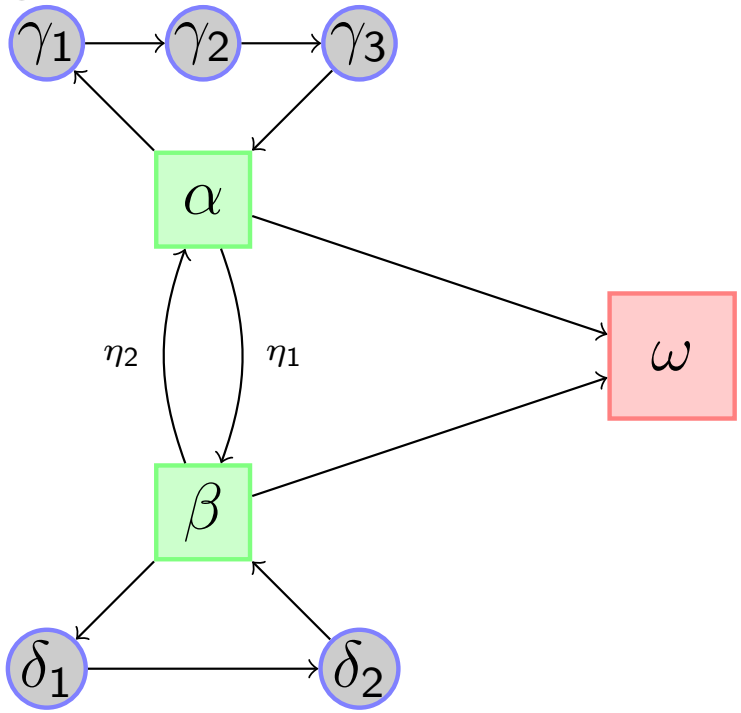
TikZ Options

- We can also give options on the line itself:
- Line widths:
  - ultra thick, very thick, thick, semithick, thin, very thin, ultra thin, help lines
  - or use "line width=12" for 12 pt, or "line width=0.2cm", or some other value and unit
- From before, the color can be determined by "draw=red", for example, or simply "red"
- Arrows can be indicated as in the first main example.
- Line style can be: dashed, dotted, solid
- Example:
  \draw[<->,dashed,ultra thick,purple] (0,0) -- (1,1) -- (2,0);

Outline

Introduction to Graphs

- For this second example, our goal is to create the following graph.



Deciding what we need

First, figuring out what the structure of our graphic is:

- There are a total of 8 nodes
  - Five nodes are in small gray circles
  - Two nodes are in medium green boxes
  - One node is in a larger red/pink box
- There are edges connecting the nodes
  - Most are straight lines
  - Two are arcs
- The arced edges also have labels on them

Nodes

- Reminder: The basic command is:
  \node at (x0,y0) {what the node says}
  but there are options available
- For our α node (green box):
  \node at (0,1) [rectangle,draw=black,fill=green] {α}
- Note that this isn't quite right. The original graph had a green outline and a lighter green interior color.
- We can blend colors in Ti*k*Z (and LaTeX).

Blending colors

- If we wanted a darker green for a background than we got, we would blend it with black.
  fill=green!50!black.
  Note: The number is the percent of the first color.
- With 50% black, it's a bit dark.
- With green!70!black it's only 30% black.
- The default second color in a blend is white, so typing
  fill=green!20
  will produce something that is 20% green and 80% white (and which is what was used in our graph).

Styles and more options

- We can adjust the formatting of each node, but...
- TikZ also allows us to create a style that can be used repeatedly
  - don't have to copy-and-paste
  - when make a change, only make it in one place
- The command to create the style looks like this

```
\begin{tikzpicture}
[nameofstyle/.style={what your style looks like}]
\end{tikzpicture}
```

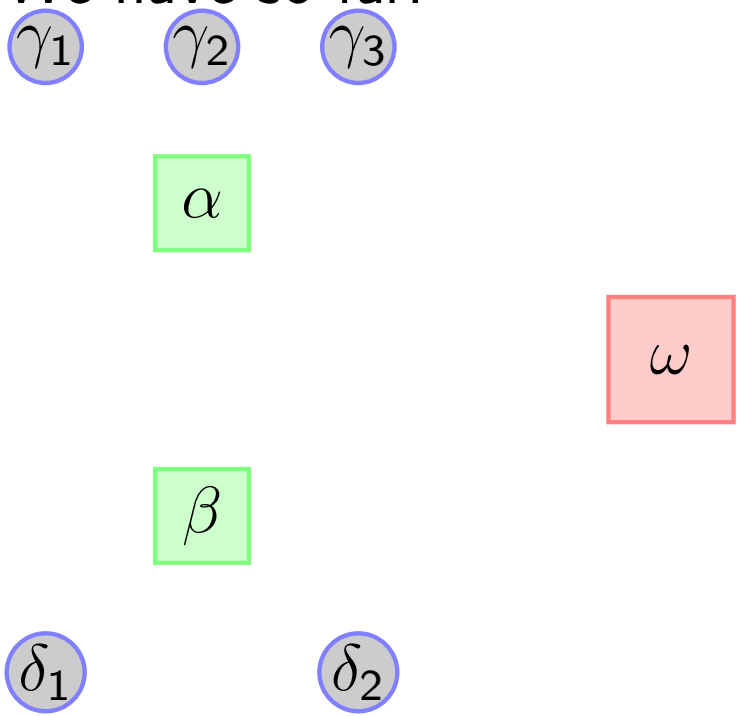
- Before we create styles, there are a couple more options of use
- `minimum size=6mm`
  creates a minimum dimension of the rectangle. This gives a uniformity to the dimensions of the nodes with this style.
- `inner sep=0pt`
  makes sure that if your minimum size is really small there will still be space for added text.

Starting our example

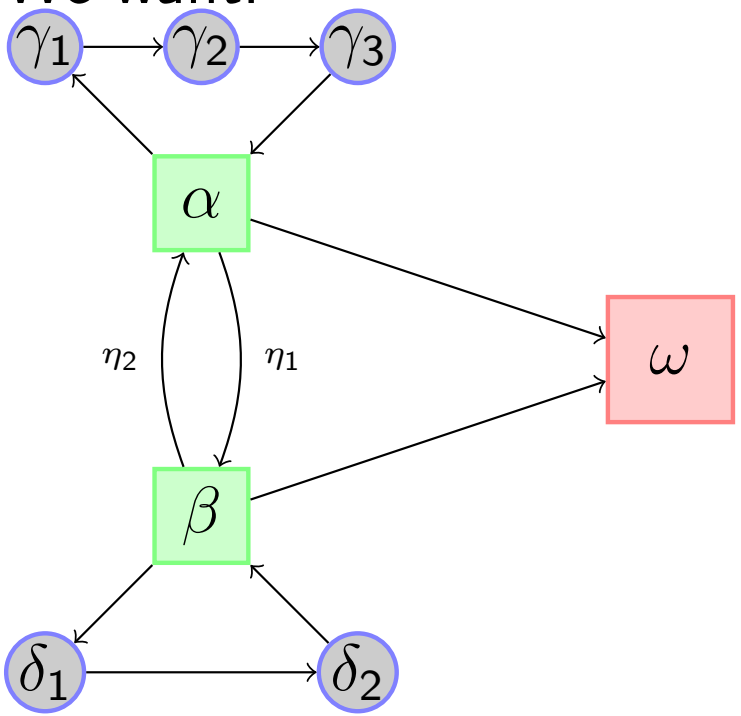
```
\begin{tikzpicture}
[source/.style={rectangle,draw=green!50,fill=green!20,thick,
inner sep=0pt,minimum size=6mm},
build/.style={circle,draw=blue!50,fill=black!20,thick,
inner sep=0pt,minimum size=4mm},
sink/.style={rectangle,draw=red!50,fill=red!20,thick,
inner sep=0pt,minimum size=8mm}]

\node (omega) at (3,0) [sink] {$\omega$};
\node (alpha) at (0,1) [source] {$\alpha$};
\node (beta) at (0,-1) [source] {$\beta$};
\node (gamma1) at (-1,2) [build] {$\gamma_1$};
\node (gamma2) at (0,2) [build] {$\gamma_2$};
\node (gamma3) at (1,2) [build] {$\gamma_3$};
\node (delta1) at (-1,-2) [build] {$\delta_1$};
\node (delta2) at (1,-2) [build] {$\delta_2$};
\end{tikzpicture}
```

Starting our example

We have so far:



We want:



So now we need to connect nodes.

Edges

- One of the standard ways to draw edges is to tell Ti*k*Z to draw an edge from one node to another, but that requires that we be able to refer to specific nodes.
- In other words, they need *names*.
- This is what the (alpha), etc., in the code was doing.
- Note that the name never appears in your final document (much like the labels that we created and referred to before, the name of the styles, etc.)
- If we change the command for gamma1 to
```
\node (gamma1) at (-1,2) [build] {$\gamma_1$} edge[<-] (alpha);
```
- This command draws an edge from (gamma1) to (alpha), but because of how we chose the arrows, it will appear as if it is going from α to γ_1.
- Note that you can't draw an edge to/from a node that does not yet exist!

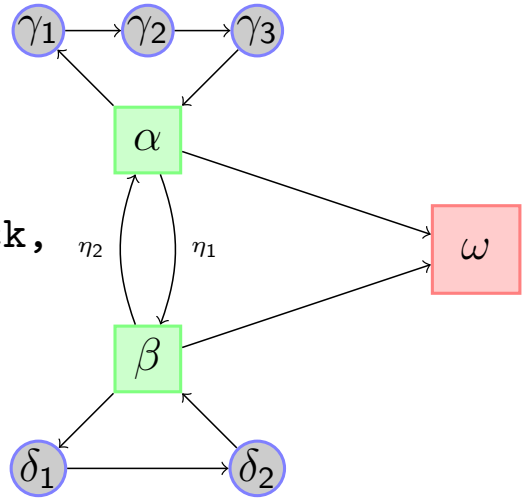
Curved edges and edge labels

- To draw the curved edges we modify the options for the edge portion of the command to
- `edge[<-,bend right=20] (alpha), or`
  `edge[->,bend left=20] (alpha), or both (our case)`
- The bend command bends the arc in the direction you want it to bend by the degrees you specify.
- You can have multiple edge commands for the same node.
- To add text to an edge, we create a node by the edge.
- We change the curved edge commands to

```
edge[<-,bend right=20] node[auto,swap] {\tiny $\eta_1$} (alpha)
edge[->,bend left=20] node[auto] {\tiny $\eta_2$} (alpha)
```

- The auto command tells ti*k*Z not to put the node right on top of the edge.
- The swap command changes the side of the edge where the node is placed.

Final Code




```
\begin{tikzpicture}
[source/.style={rectangle,draw=green!50,fill=green!20,thick,
inner sep=0pt,minimum size=6mm},
build/.style={circle,draw=blue!50,fill=black!20,thick,
inner sep=0pt,minimum size=4mm},
sink/.style={rectangle,draw=red!50,fill=red!20,thick,
inner sep=0pt,minimum size=8mm}]

\node (omega) at (3,0) [sink] {$\omega$};
\node (alpha) at (0,1) [source] {$\alpha$} edge[->] (omega);
\node (beta) at (0,-1) [source] {$\beta$} edge[->] (omega)
edge[<-,bend right=20] node[auto,swap] {\tiny $\eta_1$} (alpha)
edge[->,bend left=20] node[auto] {\tiny $\eta_2$} (alpha);
\node (gamma1) at (-1,2) [build] {$\gamma_1$} edge[<-] (alpha);
\node (gamma2) at (0,2) [build] {$\gamma_2$} edge[<-] (gamma1);
\node (gamma3) at (1,2) [build] {$\gamma_3$} edge[<-] (gamma2) edge[->] (alpha);
\node (delta1) at (-1,-2) [build] {$\delta_1$} edge[<-] (beta);
\node (delta2) at (1,-2) [build] {$\delta_2$} edge[<-] (delta1) edge[->] (beta);
\end{tikzpicture}
```

Outline

Other Programs

While Ti*k*Z is fun to use, some people don't like it because you still have to create figures using code. This is also true of pstricks. However, there are programs out there where you can "draw" pictures, save them and then import them into LaTeX.

- IPE (Windows and Mac versions exist, Free, Good for basic geometric shapes and graphs)
- Xfig (similar to IPE, Unix Based, Free)
- Winfig (Rip off of Xfig, Windows Based, Not Free)
- Maple (Good for graphing functions, not free (very expensive))
- Mathematica (Similar to Maple)
- SAGE (Free, Open Source, similar to Maple and Mathematica but a bit harder to use.)

What is IPE?

IPE stands for Integrated Picture Environment. It is a WYSIWYG (What You See is What You Get) drawing program that interacts well with LaTeX.

To download IPE, use this link and scroll down to find the appropriate binary file (Linux, Windows, or Mac OS).

Extract the files in the compressed folder to a convenient directory. The executable file for IPE version 7.2.7 can be found inside the bin folder.

You can find the IPE user manual here.

There are many drawing programs that can be used to create graphics for inclusion in a LaTeX document.