Math 291: Lecture 11

Dr. Fagerstrom

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

April 12, 2018

Outline

What is TikZ?

TikZ stands for "TikZ ist kein Zeichenprogramm", which translates to "TikZis not a drawing program".

What is TikZ?

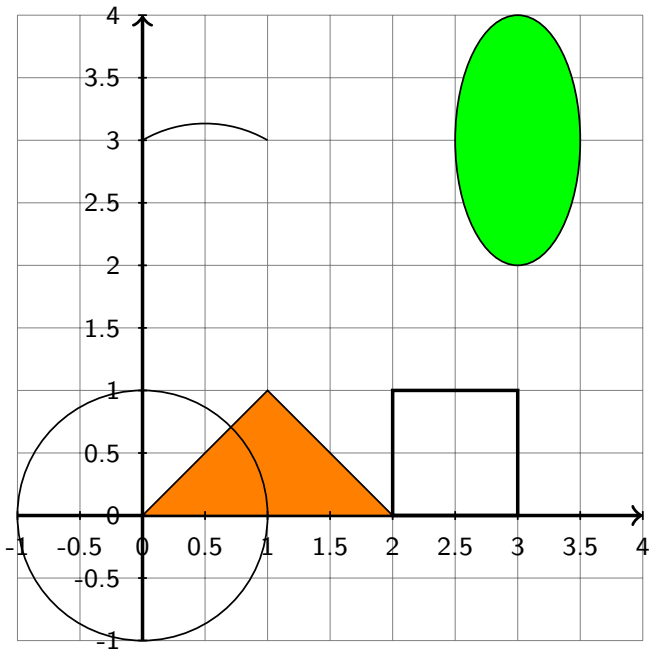
TikZ stands for "TikZ ist kein Zeichenprogramm", which translates to "TikZis not a drawing program".
But TikZ IS a drawing program with a lot of bells and whistles.

What is TikZ?

Ti*k*Z stands for "TikZ ist kein Zeichenprogramm", which translates to "Ti*k*Zis 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.

Outline

Example 1 Result



The tikZ Environment

Start a new document. In the preamble, load the package tikz. Also add the command:

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

The tikZ Environment

Start a new document. In the preamble, load the package tikz. Also add the command:

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

A tikZ diagram is created by using the tikZ environment.

The tikZ Environment

Start a new document. In the preamble, load the package tikz.
Also add the command:

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

A tikZ diagram is created by using the tikZ environment.
Add a begin and end command to your document to create the tikZ environment.

Basic Commands

- The basic commands happen within a ti*k*Z environment.
- They all end with a semicolon;
- There can be (and usually are) several commands within the same environment.
- Some basic commands are:
  - \draw (x0,y0) -- (x1,y1) -- (x2,y2);
  - \filldraw (x0,y0) -- (x1,y1) -- (x2,y2);
  - \path (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);

Practicing

- Add the following inside your ti*k*Z environment:
  \draw (0,0) -- (1,1) -- (2,0);

Practicing

- Add the following inside your ti*k*Z environment:
  \draw (0,0) -- (1,1) -- (2,0);
- Feel free to add more points and see what happens.

Practicing

- Add the following inside your ti*k*Z environment:
  \draw (0,0) -- (1,1) -- (2,0);
- Feel free to add more points and see what happens.
- Also, observe that you don't have to tell ti*k*Z right away how much space you'll need. It just makes the space it needs on its own.

Practicing

- Add the following inside your ti*k*Z environment:
  \draw (0,0) -- (1,1) -- (2,0);
- Feel free to add more points and see what happens.
- Also, observe that you don't have to tell ti*k*Z right away how much space you'll need. It just makes the space it needs on its own.
  - Note that the default unit is cm for the space that ti*k*Z sets aside per unit.
- Suppose we wanted to complete our shape to a triangle. We use the command:
  \draw (0,0) -- (1,1) -- (2,0) -- cycle;

TikZ Options

- In the picture at the beginning, the triangle was shaded orange. To shade an object we use the filldraw command.
  \filldraw[fill=orange,draw=black] (0,0) -- (1,1) -- (2,0) -- cycle;

TikZ Options

- In the picture at the beginning, the triangle was shaded orange. To shade an object we use the filldraw command.
  \filldraw[fill=orange,draw=black] (0,0) -- (1,1) -- (2,0) -- cycle;
- You should have an orange triangle with a black outline.

TikZ Options

- In the picture at the beginning, the triangle was shaded orange. To shade an object we use the filldraw command.
  \filldraw[fill=orange,draw=black] (0,0) -- (1,1) -- (2,0) -- cycle;
- You should have an orange triangle with a black outline.
- Note that the default fillcolor and drawcolor are black, and will be used unless you specify otherwise.
- Available default colors are:
  red, green, blue, cyan, magenta, yellow, black, gray, darkgray, lightgray, brown, lime, olive, orange, pink, purple, teal, violet, white

Color Gradients

- If we want to have a color gradient, we can use the `\shadedraw` command.
- Try adding

```
\shadedraw[left color=white, right color=green, draw=black]
(0,0)--(1,1)--(2,0)--cycle;
```

- This will create a color gradient in a triangle.
- 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:

TikZ Options

- We can also give options on the line itself:
- Line widths:

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

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

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"

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 with similar options as in pstricks

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 with similar options as in pstricks
- Line style can be: dashed, dotted, solid

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 with similar options as in pstricks
- Line style can be: dashed, dotted, solid
- Try the following:
  \draw[<->,dashed,ultra thick,purple] (0,0) -- (1,1) -- (2,0);

Drawing Basic Objects

Lets look at some of the other commands for drawing:

- \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

Basic Objects Practice

Add these commands in your document.

```
\draw (0,0) circle (1);
\draw (1,3) arc (60:120:1);
\draw (2,0) rectangle (3,1);
\draw (3,3) ellipse (0.5 and 1);
```

Note: You can specify the units used to measure any of the lengths: cm, pt, in, ... The default is cm.

Grid Construction

- You can quickly and easily create a coordinate grid in your picture with the following command.

```
\draw[options] (s,w) grid (n,e);
```

Grid Construction

- You can quickly and easily create a coordinate grid in your picture with the following command.

```
\draw[options] (s,w) grid (n,e);
```

- Try the following.

```
\draw[step=.5] (0,0) grid (4,4);
```

Grid Construction

- You can quickly and easily create a coordinate grid in your picture with the following command.

  ```
  \draw[options] (s,w) grid (n,e);
  ```

- Try the following.

  ```
  \draw[step=.5] (0,0) grid (4,4);
  ```

- The lines on the grid are so dark and thick that they are distracting. Try the following.

  ```
  \draw[step=.5,help lines] (0,0) grid (4,4);
  ```

Axes and Arrows

- It might be nice to have some axes on our grid, which means we need some arrows.
- Add the following commands.

```
\draw[->,thick] (0,0) -- (4,0);
\draw[->,thick](0,0) -- (0,4);
```

- Note that you get the same result with

```
\draw[<->,thick] (4,0) -- (0,0) -- (0,4);
```

- Of course, what we really want are

```
\draw[->,thick] (-1,0) -- (4,0);
\draw[->,thick](0,-1) -- (0,4);
```

Nodes: Adding Text

- In order to add text we need the next basic command in Ti*k*Z: node.
- Ti*k*Z is especially useful for drawing graphs and diagrams. (It's not very good at graphing functions.) Nodes and edges are the fundamental building blocks of a graph.

Nodes: Adding Text

- In order to add text we need the next basic command in Ti*k*Z: node.
- Ti*k*Z is especially useful for drawing graphs and diagrams. (It's not very good at graphing functions.) Nodes and edges are the fundamental building blocks of a graph.
- 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.

Nodes: Adding Text

- Add the following command to your tikzpicture.

```
\draw (0.5,1pt) -- (0.5,-1pt)
node[anchor=north] {\small 0.5};
```

Nodes: Adding Text

- Add the following command to your tikzpicture.

```
\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 (not the direction of the node from the anchor point).
- We could continue the process of labeling by adding the command:

```
\draw (1,1pt) -- (1,-1pt)
node[anchor=north] {\small 1};
```

Nodes: Adding Text

- Add the following command to your tikzpicture.

```
\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 (not the direction of the node from the anchor point).
- We could continue the process of labeling by adding the command:

```
\draw (1,1pt) -- (1,-1pt)
node[anchor=north] {\small 1};
```

- Adding all labels this way 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;
```

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
- For example, to draw the rest of our axis labels we would add the command

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

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
- For example, to draw the rest of our axis labels we would add the command

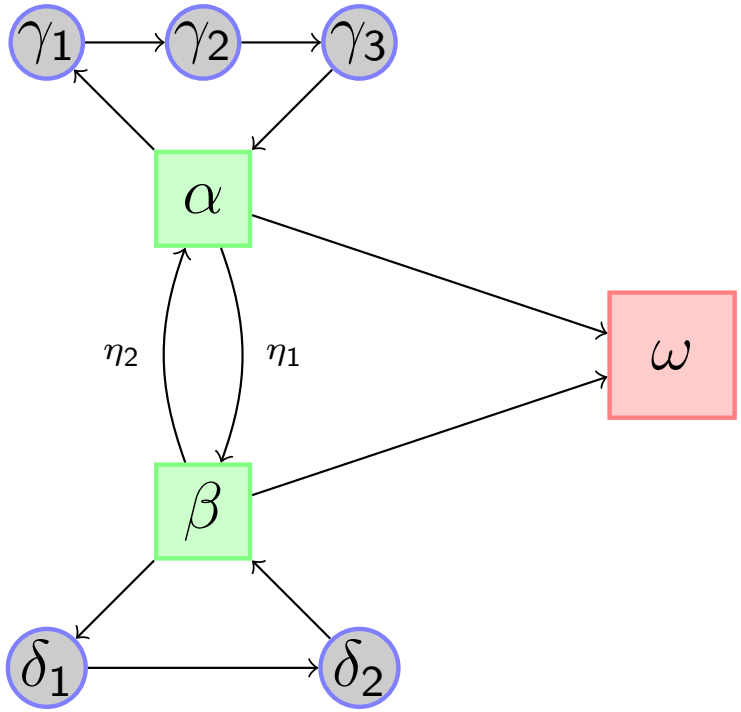
```
\foreach \x in {-1,-0.5,...,4} \draw (\x,1pt) --
(\x,-1pt) node[anchor=north]{\small \x};
```

- Try labeling the y-axis as well. (Hint: you will want to anchor your nodes in a different direction).

Outline

Introduction to Graphs

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



Using the basic node command

- Reminder: The basic command is:
  \node at (x0,y0) {what the node says}

Using the basic node command

- Reminder: The basic command is:
  \node at (x0,y0) {what the node says}
- Let's use this to create our α node at the coordinates (0,1):
  \node at (0,1) {α}

Using the basic node command

- Reminder: The basic command is:
  \node at (x0,y0) {what the node says}
- Let's use this to create our α node at the coordinates (0,1):
  \node at (0,1) {α}
- This isn't quite what we want, but there are ways of dressing it up a bit.

Using the basic node command

- Reminder: The basic command is:
  \node at (x0,y0) {what the node says}
- Let's use this to create our α node at the coordinates (0,1):
  \node at (0,1) {α}
- This isn't quite what we want, but there are ways of dressing it up a bit.
- After the location point and before the content of the node, add an optional argument:
  \node at (0,1) [rectangle,draw=black] {α}
  which tells LaTeX to put a rectangle around the contents of the node

Using the basic node command

- Reminder: The basic command is:
  \node at (x0,y0) {what the node says}
- Let's use this to create our α node at the coordinates (0,1):
  \node at (0,1) {α}
- This isn't quite what we want, but there are ways of dressing it up a bit.
- After the location point and before the content of the node, add an optional argument:
  \node at (0,1) [rectangle,draw=black] {α}
  which tells LaTeX to put a rectangle around the contents of the node
- And now add another option after the draw command, of fill=green.

Node Options

- Note that this isn't quite right. The original graph had a green outline and a lighter green interior color.

Node Options

- 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).
- 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.

Node Options

- 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).
- 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. Change it to green!70!black (so it is only 30% black).

Node Options

- 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).
- 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. Change it to green!70!black (so it is 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).

Example

- Let's add the rest of the nodes to our picture
  - ω is at (3,0)
  - α is at (0,1)
  - β is at (0,-1)
  - γ_1 is at (-1,2)
  - γ_2 is at (0,2)
  - γ_3 is at (1,2)
  - δ_1 is at (-1,-2)
  - δ_2 is at (1,-2)

Styles

- We can adjust the formatting of each node as we started with the α node, but...

Styles

- We can adjust the formatting of each node as we started with the α node, but...
- Ti*k*Z also allows us to create a style that can be used
  - don't have to copy-and-paste
  - when make a change, only make it in one place

Styles

- We can adjust the formatting of each node as we started with the α node, but...
- Ti*k*Z also allows us to create a style that can be used
  - 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}
```

Example

- Let's create the style for the α and β nodes. Add the command right after the begin tikzpicture command.

```
[source/.style={rectangle,draw=green!50,
fill=green!20,thick,inner sep=0pt,
minimum size=6mm}]
```

Example

- Let's create the style for the α and β nodes. Add the command right after the begin tikzpicture command.

  ```
  [source/.style={rectangle,draw=green!50,
  fill=green!20,thick,inner sep=0pt,
  minimum size=6mm}]
  ```

- Most of this we know, but some is new.
- `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.

Example

- Now, change your α and β nodes to

```
\node at (0,1) [source] {$\alpha$}
\node at (0,-1) [source] {$\beta$}
```

Example

- Now, change your α and β nodes to

```
\node at (0,1) [source] {$\alpha$}
\node at (0,-1) [source] {$\beta$}
```

- The other two styles will be called sink and build. Add these to the optional commands at the beginning of your picture.

```
[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}]
```

Example

- Now, change your α and β nodes to

```
\node at (0,1) [source] {$\alpha$}
\node at (0,-1) [source] {$\beta$}
```

- The other two styles will be called sink and build. Add these to the optional commands at the beginning of your picture.

```
[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}]
```

- Now, add these styles to the appropriate nodes.

Example

- We now need to connect nodes.
- 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.

Example

- We now need to connect nodes.
- 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*.

Example

- We now need to connect nodes.
- 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*.
- Change the command for α to

```
\node (alpha) at (0,1) [source] {$\alpha$}
```

- Go through and name the rest of your nodes in a similar way
- Note that the name never appears in your final document (much like the labels that we created and referred to before)

Example

- Now we can draw our edges.
- Change the command for gamma1 to

```
\node (gamma1) at (-1,2) [build]
{$\gamma_1$} edge[<-] (alpha);
```

Example

- Now we can draw our edges.
- Change the command for gamma1 to

```
\node (gamma1) at (-1,2) [build]
{$\gamma_1$} edge[<-] (alpha);
```

- This command draws an edge from α to γ_1.

Example

- Now we can draw our edges.
- Change the command for gamma1 to

```
\node (gamma1) at (-1,2) [build]
{$\gamma_1$} edge[<-] (alpha);
```

- This command draws an edge from α to γ_1.
- Note that you can't draw an edge to/from a node that does not yet exist!

Example

- Now we can draw our edges.
- Change the command for gamma1 to

```
\node (gamma1) at (-1,2) [build]
{$\gamma_1$} edge[<-] (alpha);
```

- This command draws an edge from α to γ_1.
- Note that you can't draw an edge to/from a node that does not yet exist!
- Now add all but the curved edges.

Example

- To draw the curved edges add the following command to the beta node line
- `edge[<-,bend right=20] (alpha)`
  `edge[->,bend left=20] (alpha)`
- The bend command bends the arc in the direction you want it to bend by the degrees you specify.

Example

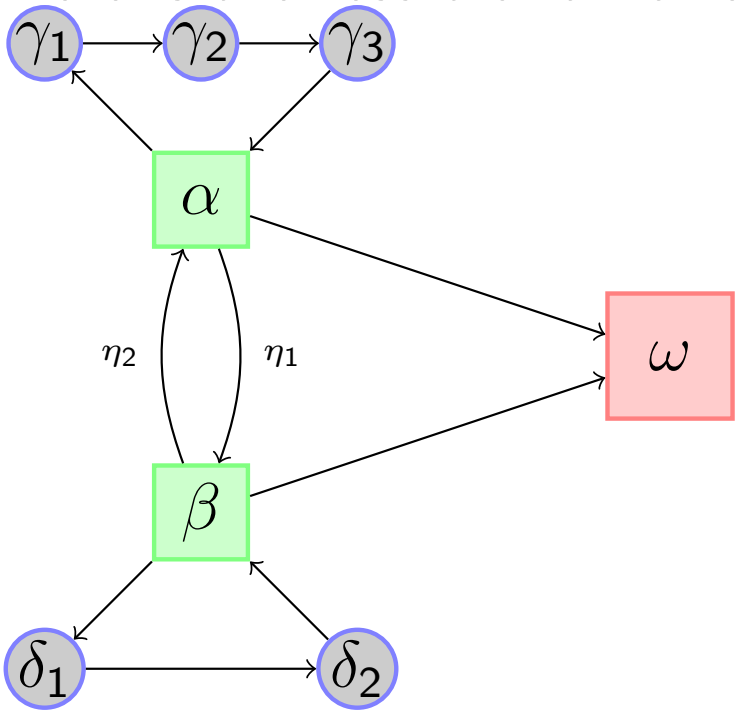
- To add text to an edge, we create a node by the edge.
- Change your curved edge commands to (on a single line)

```
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.

The Result

Here is the result of all of our hard work.



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.