CARE Tikz Reference

Tikz is a package for LaTeX used to create diagrams for all sorts of worksheets and exams, in fact, we use Tikz for **virtually** all the images/diagrams on our worksheets

Created by: Alex Garcia, July 2020

Contents

---

1 Setting up the environment

In order to use Tikz you need to import the package at the top of the .tex file

```
\usepackage{tikz}
```

The first thing you need to know about creating a 'Tikz picture' is that usually the picture should be centered, whether that's in a multicol or just on the page itself. These are the commands to create a new Tikz environment:

```
\begin{center}
    \begin{tikzpicture}
        %% image code goes here %%
    \end{tikzpicture}
\end{center}
```

2 Draw Command

This command is extremely useful and will comprise most of the drawing you will be doing in worksheet diagrams.

2.1 Basic Usage

Tikz has several basic commands, the most common one we will be using is:

```
\draw[] ;
```

Draw takes a number of arguments, most of which we will talk about later. For now, we will just focus on simple shapes and lines. **IMPORTANT NOTE**: whenever you are passing commands in the Tikz Environment don't forget to add a semicolon at the end (similar to C++ or other programming languages)

Once the new environment is set up Tikz will establish a new coordinate system for the drawing. (0,0) is the origin and it acts just like a normal Cartesian coordinate system.

In order to draw a line we need to pass two coordinates and indicate that we want a line. The command for this and result are as follows:

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

Result:

This is a line from the point (0,0) to the point (1,0) the two dashes indicate to Tikz that we want a straight line. We can do practically any number of coordinates to create a longer line, a line on an angle and even multiple lines all at once.

2.1.1 Longer line

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

Result:

2.1.2 Line at an angle

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

Result:

2.1.3 Multiple lines

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

Result:

2.2 Shapes

Another important thing that we can pass to draw is "cycle". What cycle does is it connects all the points that you passed previously, creating a full shape. Lets see what passing this argument to our multiple lines example

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

Result:

This gives us a full square without having to worry about what the last point is. Another way to create a square (or any rectangle) is to instead use 'rectangle':

2.2.1 Rectangle

```
\begin{center}
    \begin{tikzpicture}
        \draw[] (0,0) rectangle (1,1);
    \end{tikzpicture}
\end{center}
```

Result:

By giving draw two opposite corners it will create a rectangle.

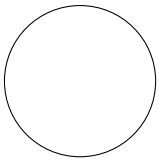
Of course, rectangles aren't the only shape that we can create easily using the draw command. We can also create circles and ellipses

2.2.2 Circle

For a circle first you pass the center point of the circle, then the word circle, then the radius

```
\draw[] (0,0) circle (1);
```

Result:

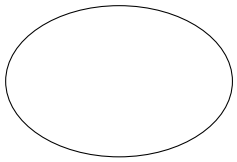


2.2.3 Ellipse

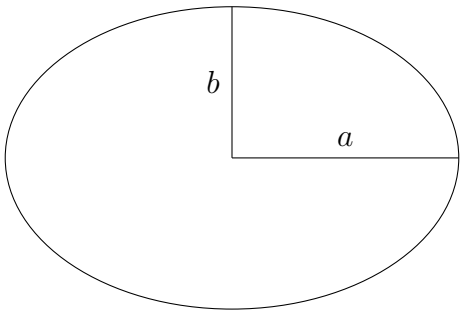
An ellipse is very similar to a circle, except instead of the radius you pass the semi-major and semi-minor axes, respectively (as well as writing ellipse instead of circle)

Result:

```
\draw[] (0,0) ellipse (1.5 and 1);
```



Quick aside on elliptical geometry:



a here is the semi-MAJOR axis while b is the semi-MINOR axis.

2.2.4 Arc

We can also break up circles and ellipses into arcs as follows:

Result:

```
\draw[] (0,0) arc (0:180:1);
```

Passing both a semi-major and semi-minor axis will give us an arc of an ellipse:

Result:

```
\draw[] (0,0) arc (0:180:1.5 and 1);
```

One thing to be wary of when creating arcs is that the position passed that was formerly the center of the shape is now the starting point for the shape (as demonstrated in the above results)

2.3 Arguments

There are a number of arguments that you can pass to draw that will each do different things.

2.3.1 Thickness

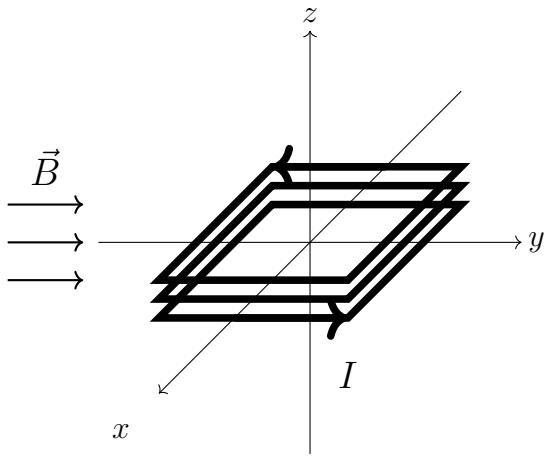
One of the primary arguments is the thickness, there are several thicknesses available:

ultra thin
very thin
thin
thick
very thick
ultra thick

In order to change the thickness pass the desired thickness into the brackets. The default thickness is thick.

```
\draw[thick] (0,0) -- (1,0);
\draw[ultra thick] (0,0) rectangle (5,4);
```

Changing the thickness of lines can come in handy in several different situations, take for example this Physics 212 question:



2.3.2 Arrows

In addition, we can make arrows with our lines. In order to do this pass one of the following into the brackets:

```
\draw[->] (0,0) -- (0,1);
\draw[<-] (1,5) -- (0,5);
\draw[<->] (-0.5,-1/3) -- (2,1/3);
```

This does the expected thing, -> is an arrow pointing right, <- is pointing left and <-> is an arrow that points both directions (You can also use |-|).

2.3.3 Dotted line

In order to make a line dotted we can pass the keyword, dashed.

```
\draw[dashed] (0,0) -- (0.2,1.5);
```

2.3.4 Color/Fill

If we want to make a line a certain color we can use the keyword color

Result:

```
\draw[color=red] (0,0) -- (3,0);
```


Similarly, we can make a shape and then fill it with a certain color using fill

(Tikz provides the colors white, black, red, green, blue, cyan, magenta and yellow)

Result:

```
\draw[fill=green] (0,0) rectangle (3,1);
```

If we want the border to be the same color as the rest of the drawing we can use the keyword draw and set it equal to the same color or none

Result:

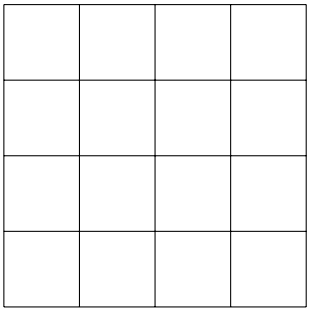
```
\draw[fill=green,draw=none] (0,0)
    rectangle (3,1);
\draw[fill=green,draw=green] (0,0)
    rectangle (3,1);
```

2.3.5 Grid

One particular 'shape' we neglected to mention previously was a grid. We can create a grid in a similar way as creating a rectangle:

Result:

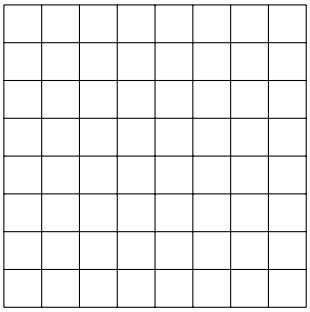
```
\draw[] (0,0) grid (4,4);
```



Again, similar to rectangle the two points are opposite corners of the grid. The reason we didn't include grid in the shapes section was that passing the argument step we can set the width of the boxes the grid creates

Result:

```
\draw[step=0.5] (0,0) grid (4,4);
```



2.3.6 Plotting

Similar to grid, plotting is a shape-like command that require a couple arguments to make logical sense.

```
\draw[samples=number,domain=from:to,smooth,variable=\x]
    plot ({\x},{%function%});
```

There's a lot to unpack here, so let's start at the beginning.

The samples is the number of points draw will plot, the default number is very low so pick a decent amount of points. With that being said... putting too many points on too many plots can slow down your compiling speed.

Domain is on what interval you want to plot the function. You'll pass the lower bound, semicolon, upper bound.

Smooth is just something we add to make the function... smooth.

The variable is just (for the most part) going to be x. You can use whatever letter you want as long as you're consistent throughout your function.

Inside the parentheses of the plot function we have the two expressions for the two coordinates for every point. Tikz will interpret your function at however many samples you pass it.

2.3.7 Scale

Using the scale keyword you can make drawings larger or smaller. Scale of greater than 1 increases the size while less than 1 decreases the size.

```
\draw[scale=0.5] (0,0) rectangle (2,4);
\draw[scale=2] (0,0) rectangle (0.5,1);
```

2.3.8 Using multiple arguments

As you might expect, using more than one argument in the brackets is totally allowed. Just separate each keyword by a comma

Result:

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

3 Filldraw Command

Similar to draw, filldraw makes shapes.

```
\filldraw[] ;
```

3.1 Arguments

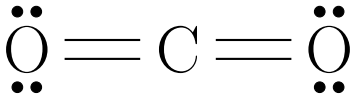
Filldraw takes pretty much the same arguments as draw. The most useful feature is that you don't have to change the outline color

Result:

```
\filldraw[color=green]
    (0,0) rectangle (3,1);
```

3.2 Dots

One of the most useful things to use filldraw for is the dots, take this Chem 102 Lewis Diagram for example:



These dots around the Oxygen atoms are done with filldraw. The way to create these dots is the following:

```
\filldraw[] (%location%) circle (%Size in pts%);
```

Here is a quick reference of point sizes

• 1pt • 2pt ● 3pt ● 4pt

4 Nodes

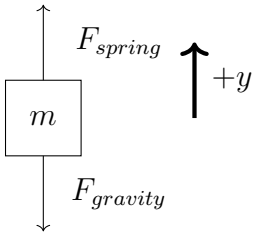
In the context of CARE and our basic, crude diagrams, nodes are just our way of adding labels. Using a node you can add text (or math mode expressions) anywhere you want on the diagram or associated with some draw or filldraw.

```
\node[] at (%location%) {%Name of Node%};
\filldraw[] (0,0) circle (1pt) node[anchor=west] {$v_{b,r}$};
```

4.1 Basics of Nodes

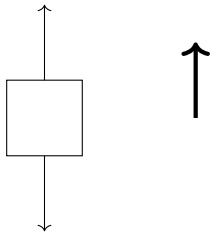
As mentioned before we are just using nodes for labels for our diagrams. For a more complete description of nodes and more interesting and fun things to do with them please visit this website

However, we will go into some detail here about how we use nodes in our diagrams. Let's do a worked example to understand, this free body diagram is from the solutions to our Physics 100 Worksheet:



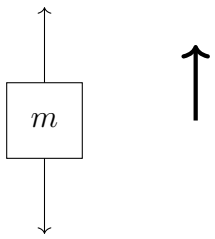
From previous sections we know we can build the skeleton of this free body diagram and the arrow for our coordinate system label.

```
% Box
\draw[] (1,0) -- (1,1) -- (0,1) -- (0,0) -- cycle;

% Vectors
\draw[->] (0.5,1) -- (0.5,2);
\draw[->] (0.5,0) -- (0.5,-1);

% Axis
\draw[->, ultra thick] (2.5,0.5) -- (2.5,1.5);
```



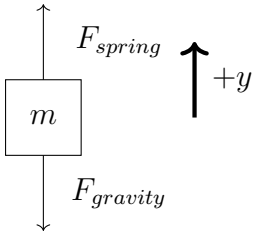
Now we want to add labels to this diagram we can use the node function to do this. We want to label the box as the mass m. To do this we need to find the center of the box, in this case (0.5,0.5) so we add this command to the picture:

```
\node[] at (0.5,0.5) {$m$};
```



We can repeat this process for each of the vectors representing the forces. We find the middle of the arrows and then place the text a little to the right of them so they don't overlap with the arrows. Then we do the same for the arrow representing the coordinate system.

```
\node[] at (1.5,1.5) {$F_{spring}$};
\node[] at (1.5,-0.5) {$F_{gravity}$};
\node[] at (3,1) {$+y$};
```



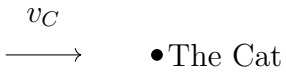
This general process is how we created ***virtually*** all of our labels for our diagrams and is a very quick and dirty way of labeling your diagrams with Tikz.

4.2 Nodes within another expression

The other way that we can add nodes/labels to our diagrams is in our filldraw or draw commands. This is demonstrated below:

```
\draw[->] (-1,0) -- (0,0) node[] at (-0.5,0.5) {$v_C$}
\filldraw[] (1,0) circle (2pt) node[anchor=west] {The Cat};
```

Which produces the following



4.2.1 Anchor

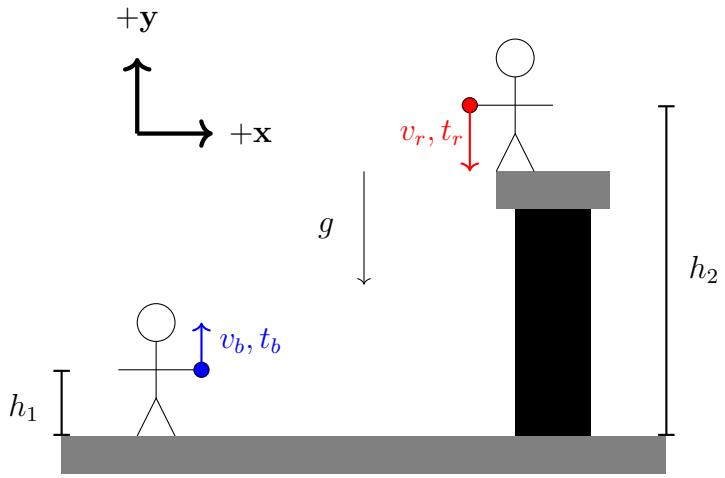
You've probably seen us use this anchor argument in this section and wondered what it was used for.

The keyword anchor just places the text in a convenient place for us, without the hassle of having to find a coordinate. There are several valid expressions for this argument:

Notice that the anchors are the OPPOSITE of the cardinal direction they normally represent

All the following also work (with the same convention): north west, north east, south east and south west.

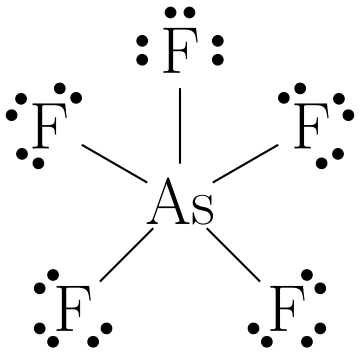
Congratulations! If you made it this far through the document you have learned a great deal about Tikz and what we do with it in CARE. You now have enough information and skills to create beautiful, high-quality diagrams like this one from our Physics 100 worksheet:



For most of the coding and diagram-making you'll be doing within Math and Physics the document up until here is really all you'll need. The next couple of sections are more specializations for specific tasks. The last section is on being neat in your tikz pictures and should be over viewed in order for your diagrams to be editable by anyone at any time.

5 Scope

Scope is a critical tool for a lot of diagrams you could create. Take this Lewis Diagram from our Chem 102 worksheet solutions:



For reference this code is about 60 lines for this diagram. We could make it less but it would require way more thought on our end.

Look at each of the electrons attached to the Fluorine's. Thinking about how to place them, our naive guess would be to figure out each coordinate and then place them accordingly. This could be less writing of code by WAY more work on our part. What we have Tikz do here is rotate the entire bond-Flourine-electrons system for us, saving us a great deal of work.

5.1 Creating a scope

Before we dive into the deep-end anymore with this chemistry example, let's back up and explain what a scope is and how to make one.

A scope is some small part of a diagram that we want to translate/shift/rotate to some other position on the diagram without having to think too hard.

Creating a scope in our code is relatively simple:

```
\begin{center}
    \begin{tikzpicture}
        \begin{scope}[% arguments]
            %% image code goes here%%
        \end{scope}
        %% other image code may go here as well %%
    \end{tikzpicture}
\end{center}
```

5.1.1 Arguments for scope

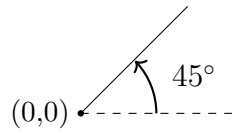
There are a couple different things we usually pass for an argument for a scope:

```
\begin{scope}[rotate around={angle:pivot}]
    %% image code goes here %%
\end{scope}
\begin{scope}[shift={amount to move it over}]
    %% image code goes here %%
\end{scope}
```

Rotate around does very much what you would expect it to do, it takes all the things in our scope and rotates it around your specified pivot. Here's an example

```
\begin{scope}[shift around={45:(0,0)}]
    \draw[] (0,0) -- (2,0);
\end{scope}
```

Result:

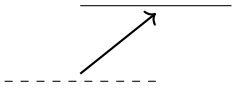


Important note: rotate-around does not rotate text, which is why our Fluorine's aren't at a weird angle.

Similar to rotate around, shift does the predictable thing. It takes everything in your scope and shifts it by your specified input. Here's an example:

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

Result:



6 Circuitikz

Circuitikz is an extremely similar package to regular-old tikz. It can be imported using:

```
\usepackage{circuitikz}
```

This package, unsurprisingly, is more geared towards circuits and creating circuit diagrams.

Creating the environment for cicruitikz is extremely similar to tikz:

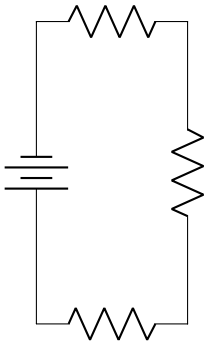
```
\begin{center}
    \begin{circutikz}
        %% image code goes here %%
    \end{circuitikz}
\end{center}
```

6.1 Draw function

Our good-old friend draw function is back. This time he has some easier functionality and can really help us draw an efficient circuit

Draw is called in the same way as before but now instead of having to use hyphens we can use to[element]. This is best described in an example:

```
\draw[] (0,0) to[battery] (0,4) to[R]
    (2,4) to[R] (2,0) to[R] (0,0);
```

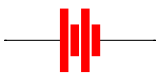


Inside the brackets of the to[ ] we put the element of the circuit that we want to connect thru the points. A complete list of elements available can be found here.

Note that the size, thickness and color of each element is fixed at some default. Changing these can be done with the color and thickness keywords inside the to[ ]

Result:

```
\draw[] (0,0)
    to[battery,color=red,ultra thick]
    (2,0);
```



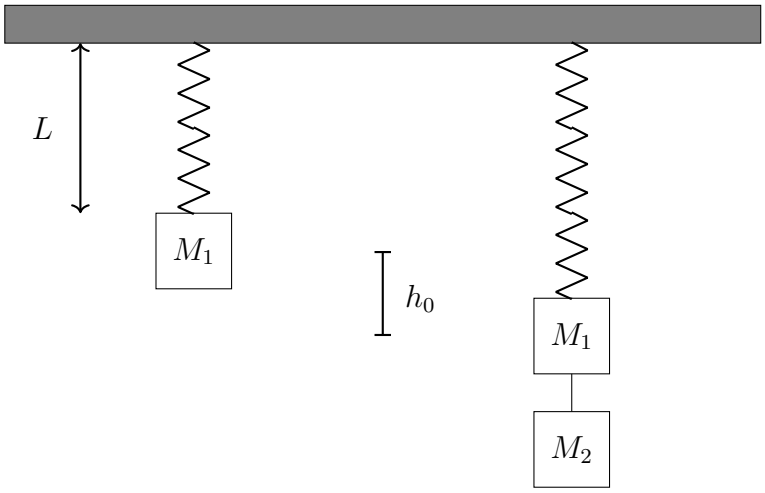
This makes circuits far easier and far cleaner to code, trust us, we've tried the other way.

6.2 Using resistors as springs

Yes, you read that right. Oftentimes to create springs we will just create our diagram using circuitikz. This is much easier than creating a spring from scratch.

Here's an example from our Physics 211 Worksheet

```
% Ceiling
\draw[fill=gray] (-5,0)
    rectangle (5,0.5);

% Springs
\draw (-2.5,0) to[R]
    (-2.5,-1.125) to[R]
    (-2.5,-2.25);
\draw (2.5,0) to[R]
    (2.5,-1.125) to[R]
    (2.5,-2.25) to[R]
    (2.5,-3.375);

... etc ...
```



7 Cleanliness of Code

In order for your Tikz picture code to be readable by others there are some common practices that are really useful.

7.1 Labeling

Labeling everything in your code is incredibly important, especially as you're making more and more complex diagrams.

You may have already noticed in the some of the earlier snippets of code our system for labeling. We'll reuse the same snippet from our free body diagram before:

```
% Box
\draw[] (1,0) -- (1,1) -- (0,1) -- (0,0) -- cycle;

% Vectors
\draw[->] (0.5,1) -- (0.5,2);
\draw[->] (0.5,0) -- (0.5,-1);

% Axis
\draw[->, ultra thick] (2.5,0.5) -- (2.5,1.5);
```

Here we have each part of our diagram that we're labeling having it's own little label. Employing this system helps who ever is reading your code understand exactly what you are creating with each line or couple of lines.

This might seem like a super insignificant detail when you look at smaller diagrams like this one, but as you're creating more and more complex diagrams (60,70,... 100 lines of code) it becomes increasingly important to make sure you're making your work readable.

7.2 Spacing and Ordering

Does this code look familiar? Can you tell what we are trying to make?

```
\draw[->,ultra thick](2.5,0.5)--(2.5,1.5);
\draw[->](0.5,1)--(0.5,2);
\draw[](1,0)--(1,1)--(0,1)--(0,0)--cycle;
\draw[->](0.5,0)--(0.5,-1);
```

It's the same code as before, just in a random order and without any spacing.

Along with labeling, spacing is critical in being able to read and picture a diagram without actually compiling your code. In addition, it doesn't matter how well everything is spaced out if you don't have everything in order and like pieces together.

7.3 Tabing

Another good habit to get yourself into is hitting the Tab key on your keyboard.

Whenever we use a new begin command or start new code inside of something it is good, common courtesy to tab (Not everyone does to everything this but in order to stick with our convention please do).

Example of good, clean code:

```
\begin{center}
    \begin{tikzpicture}
        % Stick being thrown
        \begin{scope}[rotate around={-125:(10,0)}]
            \draw[] (0,0) -- (1,0);
        \end{scope}

        % Conveyor Belt
        \draw[->,thick] (0,0) arc (180:0:4 and 0.75);
    \end{tikzpicture}
\end{center}
```

Interested in seeing all the diagram on all the CARE worksheets?