

A Tikz Tutorial

Felix Zhou

May 10, 2019

Contents

1	Generic Setup	5
2	Paths	6
2.1	Straight Paths	6
2.2	Curved Paths	6
2.2.1	Bezier Curves and the Bernstein Polynomial	7
2.3	Circular Paths	7
2.4	Rectangular Paths	8
2.5	Grid Paths	9
2.6	Arc Paths	9
2.7	Clipping	10
2.8	Polar Coordinates	11
2.9	Relative Coordinates	11
2.10	Arrows	13
3	Graphs	14
3.1	Nodes	14
3.2	Node Sizes	14
3.3	Relative Positioning	15
3.4	Labels	15
3.5	Connecting Nodes	16
3.6	Snake Edges	16
4	Options	18
4.1	Styles	18
4.2	Thickness	19
4.3	Scaling	19
4.4	Scoping	20
4.5	Transformations	20
4.6	For Loops	21
4.7	Layers	21

List of Figures

1	Basic Straight Path	6
2	Inlined Straight Path	6
3	Basic Curve	6
4	Basic Circle	8
5	Basic Ellipse	8
6	Basic Rectangle	8
7	Basic Grid	9
8	Basic Circle Arc	9
9	Basic Ellipse Arc	9

10	Arc Line Intersection	10
11	Rectangular Clip	10
12	Ellipse Clip	11
13	Extended Angle	11
14	Basic Relative Coordinetes	12
15	Set Relative Coordinetes	12
16	Simple Arrow	13
17	Basic Nodes	14
18	Basic Sized Nodes	14
19	Relative Named Nodes	15
20	Basic Labeled Nodes	15
21	Basic Connected Nodes	16
22	Basic Snake Edge	16
23	Adjusted Snake Edge	16
24	Basic Styles	18
25	Modified Style	18
26	Thickness	19
27	Small Circle	19
28	Scaled Circle	20
29	Basic Scoped Lines	20
30	Basic Transformations	20
31	Basic Repetitions	21
32	Basic Background Layer	21

List of Tables

Introduction

What it this?

I am an upcoming third year (3A) student at the University of Waterloo who plans to take CO342 - Introduction to Graph Theory, in Fall 2019. To aid me and other students such as myself with typesetting graphs using tikz, I created this shortened tutorial from the TikZ and PGF Manual. Please do not hesitate to reach out to me if there are any errors and I will publish an updated version ASAP.

Goal

I have always believed in understanding your tools completely before using them. In addition to my tutorial on TikZ, I will include as much notes on the math behind non-trivial TikZ commands as I can, with the condition that I myself at least understand the gist of the technologies. If it is completey above me and looks like goby goop, I will clearly note this and invite *you* to write me a short section based on your understanding.

1 Generic Setup

```
\documentclass{article}
\usepackage{tikz}
\usepackage{float}

\begin{document}
\begin{figure}[H] % forces position of tikz to be where it is in source file
  \centering

  \begin{tikzpicture}
    % insert provided code
  \end{tikzpicture}

  \caption{A Caption}
  \label{fig:alabel}
\end{figure}
\end{document}
```

This is our default setup for any document classes involving TikZ. If additional imports are necessary, we will explicitly note them.

Note that we may omit typing “tikzpicture” everytime and inline simple TikZ pictures with the “tikz” command. When we do this, we will explicitly type out “tikz”.

If an option is enabled for the entire tikzpicture environment, we will include the begin and end keywords explicitly.

2 Paths

2.1 Straight Paths

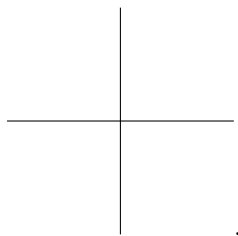


Figure 1: Basic Straight Path

```
\draw (-1.5,0) -- (1.5,0); % draw straight lines between two points
\draw (0,-1.5) -- (0,1.5);
```

We can also directly inline a similar picture

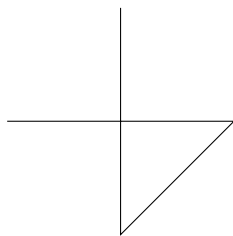


Figure 2: Inlined Straight Path

```
\tikz \draw (-1.5,0) -- (1.5,0) -- (0,-1.5) -- (0,1.5);
```

2.2 Curved Paths

TikZ allows us to define arbitrary curves using control points. The basis behind of this concept are Bézier Curves, which in turn are based on the Bernstein Polynomial.

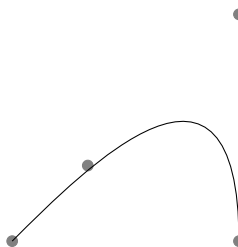


Figure 3: Basic Curve

```

\filldraw [gray]
(0,0) circle (2pt)
(1,1) circle (2pt)
(3,3) circle (2pt)
(3,0) circle (2pt);
\draw (0,0) .. controls (1,1) and (3,3) .. (3,0);

```

The above shows a basic curve where the control points are explicitly drawn as well. Obviously, this would be omitted (on your graph theory course home work for example).

2.2.1 Bézier Curves and the Bernstein Polynomial

This section gives some mathematical background on curved paths and is skippable.

Definition 2.2.1 (Bézier Curve)

A recursive definition for the Bézier curve of degree n expresses it as a point-to-point linear combination (interpolation) of a pair of corresponding points in two Bézier curves of degree $n - 1$.

$$\begin{aligned}
 B : \mathbb{R} &\rightarrow \mathbb{R} \\
 B_{P_0}(t) &= P_0 && \text{base case} \\
 B(t) &= B_{P_0 P_1 \dots P_n}(t) = (1 - t)B_{P_0 P_1 \dots P_{n-1}}(t) + tB_{P_1 P_2 \dots P_n}(t)
 \end{aligned}$$

Definition 2.2.2 (Bernstein Basis Polynomials)

of degree n are given by

$$\left\{ b_{i,n}(t) = \binom{n}{i} t^i (1 - t)^{n-i} : 0 \leq i \leq n \right\}$$

which form a basis of polynomials with degree at most n .

Proposition 2.2.1

$$B(t) = \sum_{i=0}^n \binom{n}{i} t^i (1 - t)^{n-i} P_i$$

So we can express a Bézier curve as a linear combination of the Bernstein Basis.

The points P_i are called *Control Points* for the Bézier Curve. The polygon formed by connecting the Bézier points with lines, starting with P_0 and finishing with P_n , gives the *Bézier Polygon (Control Polygon)*. The convex hull of the Bézier Polygon contains the Bézier Curve.

2.3 Circular Paths

Although it would certainly be possible to draw all our circular paths with control points, it would prove tedious to say the least.

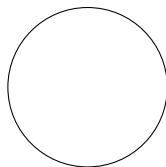


Figure 4: Basic Circle

```
\tikz \draw (0,0) circle (2);
```

We can also draw Ellipses.

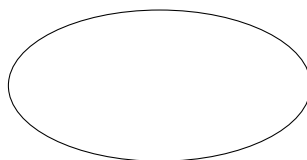


Figure 5: Basic Ellipse

```
\tikz \draw (0,0) ellipse (2 and 1);
```

Although it is possible to draw ellipses which are rotated in arbitrary directions, we will leave this for the section on transformations later on.

2.4 Rectangular Paths



Figure 6: Basic Rectangle

```
\tikz \draw (0,0) rectangle (4, 3);
```


2.5 Grid Paths

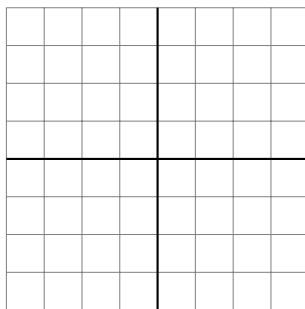


Figure 7: Basic Grid

```
\draw[step=.5, gray, very thin] (-2,-2) grid (2, 2);  
\draw[thick] (-2, 0) -- (2, 0);  
\draw[thick] (0, -2) -- (0, 2);
```

Note how we indicated the step and, color, and thickness of the paths to TikZ to accentuate the x, y-axis instead of the entire grid.

2.6 Arc Paths

What if we only wish to draw part of a circle or ellipse? TikZ has us covered. The second argument to the arc command consists of the form (initial angle:final angle:radius/radii), where the angles are given in degrees and interpreted in a counter clockwise fashion, not unlike the unit circle and the sinusoidal functions.



Figure 8: Basic Circle Arc

```
\tikz \draw (0, 0) arc (0:90:1);
```

We can actually give TWO arguments to the third parameter of the second argument to arc, which will draw an ellipse.

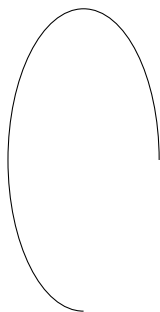


Figure 9: Basic Ellipse Arc

```
\tikz \draw (0, 0) arc (0:270:1 and 2);
```

2.7 Clipping

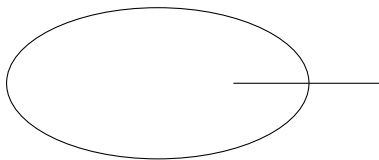


Figure 10: Arc Line Intersection

```
\draw (1, 0) arc (0:360:2 and 1);  
\draw (0, 0) -- (2, 0);
```

Consider the above. Suppose we wish to emphasize the point of intersection of the arc and the line. We can use the “clip” command to crop the rendering.

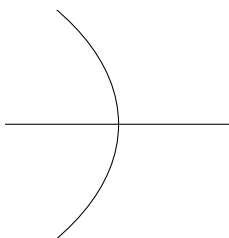


Figure 11: Rectangular Clip

```
\begin{tikzpicture}[scale=3]  
  \clip (0.5, -0.5) rectangle (1.5, 0.5);  
  \draw (1, 0) arc (0:360:2 and 1);  
  \draw (0, 0) -- (2, 0);  
\end{tikzpicture}
```

However, we are not just limited to rectangular clips! Below shows the example above with some augmentations, an ellipse-shaped clip, as well as grid lines added.

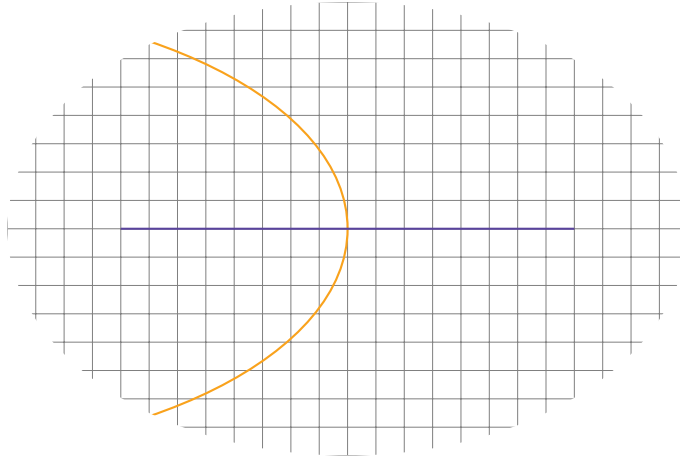


Figure 12: Ellipse Clip

```
\begin{tikzpicture}[scale=3]
  \clip (1, 0) ellipse (1.5 and 1);
  \draw[step=.125, very thin, gray] (-2, -2) grid (3, 2);
  \draw[thick, color=YellowOrange] (-1, -1) arc (-90:90:2 and 1);
  \draw[thick, color=Violet] (0, 0) -- (2, 0);
\end{tikzpicture}
```

2.8 Polar Coordinates

We have been previously only using Cartesian Coordinates up until this point. However, it may be convenient to represent information using Polar Coordinates in the future.

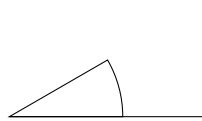


Figure 13: Extended Angle

```
\begin{tikzpicture}[scale=3]
  \draw (0, 0) -- (0.5, 0) arc (0:30:0.5) -- cycle;
  \draw (30:1cm) |- (0,0);
\end{tikzpicture}
```

Note how we easily extended the angle using Polar Coordinates. Also note the new “—” syntax which asks TikZ to draw a vertical line through the first argument intersection a horizontal line through the second argument. We will condense more new syntax this way in the future as it does add completely new tools to our arsenal but is good to know for niche situations such as the above.

2.9 Relative Coordinates

By default, Cartesian Coordinates are scaled to the previously defined vector, with $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ being the default vector. We can ask for TikZ to draw points *relative* to the previous vector using the +, ++ syntax

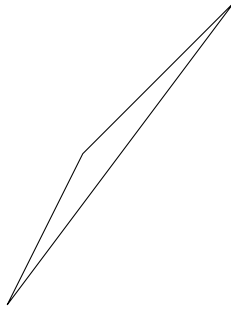


Figure 14: Basic Relative Coordinates

```
\tikz \draw (0, 0) -- +(1, 2) -- +(3, 4) -- cycle;
```

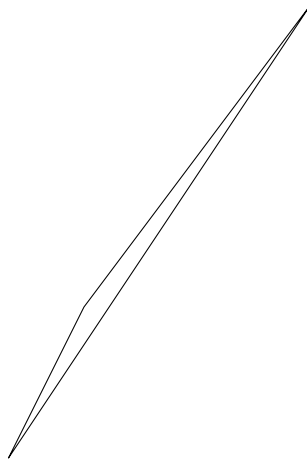


Figure 15: Set Relative Coordinates

```
\tikz \draw (0, 0) -- ++(1, 2) -- ++(3, 4) -- cycle;
```

Note the difference between the two, the “+” syntax is a scalar addition to the previous vector but does **not** mutate the previous vector in any way. On the other hand, “++” has the same value as “+” but also mutates the previous vector to be the point to specified with the “++” syntax.

Both are plausible and no one is superior to the other in all cases. We advise you to know your entire toolbox and choose the correct coordinate system when the time is right.

2.10 Arrows

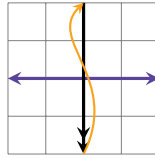


Figure 16: Simple Arrow

```
\begin{tikzpicture}[>=stealth]
  \draw[step=.5,color=gray,very thin] (-1, -1) grid (1, 1);
  \draw[<->, very thick, Violet] (-1, 0) -- (1, 0);
  \draw[<<->, very thick] (0, -1) -- (0, 1);
  \draw[->, thick, YellowOrange] (0, -1) .. controls (0.5, 0) and (-0.5, 0.5) .. (0, 1);
\end{tikzpicture}
```

As a rule of thumb, we can only draw arrows on paths that are open in some sense. Note that we can indicate the kind of arrow head desired using the “>” option in the environment setting.

3 Graphs

This is definitely my favourite part of TikZ.

3.1 Nodes

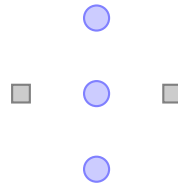


Figure 17: Basic Nodes

```
\tikzstyle{place}=[circle,draw=blue!50,fill=blue!20,thick]
\tikzstyle{transition}=[rectangle,draw=black!50,fill=black!20,thick]
\begin{tikzpicture}
  \node at ( 0,2) [place] {};
  \node at ( 0,1) [place] {};
  \node at ( 0,0) [place] {};
  \node at ( 1,1) [transition] {};
  \node at (-1,1) [transition] {};
\end{tikzpicture}
```

3.2 Node Sizes

By default, nodes have non-zero sizes as TikZ has default non-zero inner sep between text and node boundaries. We set this to 0 and instead use the minimum size command which allows the nodes to shrink to a minimum size when there is no text and expand as needed.

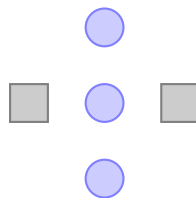


Figure 18: Basic Sized Nodes

```
\tikzstyle{place}=[circle,draw=blue!50,fill=blue!20,thick,inner sep=0,minimum size=0.5cm]
\tikzstyle{transition}=[rectangle,draw=black!50,fill=black!20,thick,inner sep=0,minimum size=0.5cm]
\begin{tikzpicture}
  \node at ( 0,2) [place] {};
  \node at ( 0,1) [place] {};
  \node at ( 0,0) [place] {};
  \node at ( 1,1) [transition] {};
  \node at (-1,1) [transition] {};
\end{tikzpicture}
```

3.3 Relative Positioning

First we name the nodes, then describe their position relative to one another, the value node distance is important in our case.

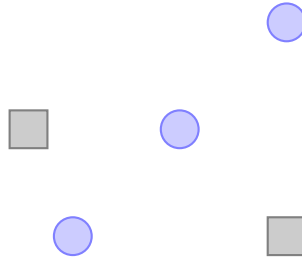


Figure 19: Relative Named Nodes

```
\tikzstyle{place}=[circle,draw=blue!50,fill=blue!20,thick,inner sep=0,minimum size=0.5cm]
\tikzstyle{transition}=[rectangle,draw=black!50,fill=black!20,thick,inner sep=0,minimum size=0.5cm]
\begin{tikzpicture}[node distance=2cm]
  \node[place] (waiting) {};
  \node[place] (critical) [below left of=waiting] {};
  \node[place] (semaphore) [below left of=critical] {};
  \node[transition] (leave critical) [below right of=critical] {};
  \node[transition] (enter critical) [left of=critical] {};
\end{tikzpicture}
```

3.4 Labels

We can always explicitly define blank nodes as labels but the label option is much better suited to our needs.

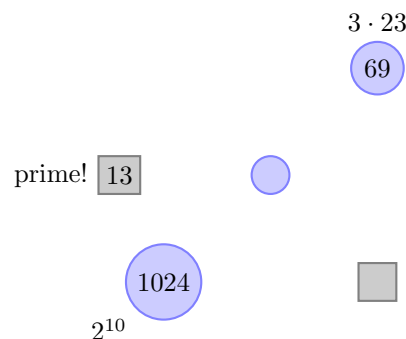


Figure 20: Basic Labeled Nodes

```
\tikzstyle{place}=[circle,draw=blue!50,fill=blue!20,thick,inner sep=0.1cm,minimum size=0.5cm]
\tikzstyle{transition}=[rectangle,draw=black!50,fill=black!20,thick,inner sep=0.1cm,minimum size=0.5cm]
\begin{tikzpicture}[node distance=2cm]
  \node[place] (waiting) [label=above:$3\cdot 23$] {$69$};
  \node[place] (critical) [below left of=waiting] {};
  \node[place] (semaphore) [below left of=critical,label=below left:$2^{10}$] {$1024$};
  \node[transition] (leave critical) [below right of=critical] {};
  \node[transition] (enter critical) [left of=critical,label=left:prime!] {$13$};
\end{tikzpicture}
```

3.5 Connecting Nodes

Here comes the fun part!

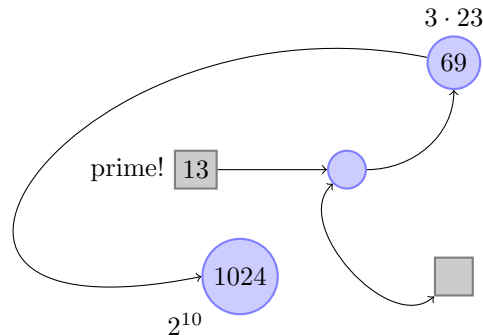


Figure 21: Basic Connected Nodes

```
\tikzstyle{place}=[circle,draw=blue!50,fill=blue!20,thick,inner sep=0.1cm,minimum size=0.5cm]
\tikzstyle{transition}=[rectangle,draw=black!50,fill=black!20,thick,inner sep=0.1cm,minimum size=0.5cm]
\begin{tikzpicture}[node distance=2cm]
  \node[place] (waiting) [label=above:$3\cdot 23$] {$69$};
  \node[place] (critical) [below left of=waiting] {};
  \node[place] (semaphore) [below left of=critical,label=below left:$2^{10}$] {$1024$};
  \node[transition] (leave critical) [below right of=critical] {};
  \node[transition] (enter critical) [left of=critical,label=left:prime!] {$13$};
  \draw[->] (enter critical) -- (critical.west);
  \draw[->] (waiting) .. controls +(-5, 1) and +(-5, -1) .. (semaphore.west);
  \draw[<-] (waiting) to [out=270,in=0] (critical);
  \draw[<->] (critical) to [bend right=90] (leave critical);
\end{tikzpicture}
```

Here we showcase three of the many ways to connect nodes: through explicit paths, through indicating exit points, and through the bend command.

3.6 Snake Edges



Figure 22: Basic Snake Edge

```
\usetikzlibrary{decorations.pathmorphing}
\tikz \draw[->,decorate,decoration={snake}] (0, 0) -- (10, 0);
```

We can control the behavior of the edge like below



Figure 23: Adjusted Snake Edge


```
\usetikzlibrary{decorations.pathmorphing}  
\tikz \draw[->,decorate,decoration={snake,amplitude=.4mm,segment length=2mm}] (0, 0) -- (10, 0);
```

4 Options

4.1 Styles

We now go on a slight detour. If you came from any sort of programming background, you would know the importance of avoiding code duplication. TikZ gives us the ability to avoid passing the same options to the “draw” command by defining “styles”.

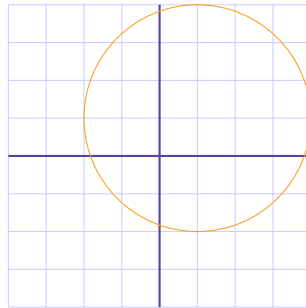


Figure 24: Basic Styles

```
\draw[step=.5, color=white!80!blue, very thin] (-2, -2) grid (2, 2);  
\draw[thick, color=Violet] (-2, 0) -- (2, 0);  
\draw[thick, color=Violet] (0, -2) -- (0, 2);  
\draw[color=YellowOrange] (0.5, 0.5) circle (1.5);
```

The below is equivalent code, but much more extensible and modular.

```
\tikzstyle Grid Line=[step=.5, color=white!80!blue, very thin]  
\tikzstyle Axis Line=[thick, color=Violet]  
\tikzstyle Circle Line=[color=YellowOrange]  
  
\draw[style=Grid Line] (-2, -2) grid (2, 2);  
\draw[style=Axis Line] (-2, 0) -- (2, 0);  
\draw[Axis Line] (0, -2) -- (0, 2);  
\draw[Circle Line] (0.5, 0.5) circle (1.5);
```

Note that the “style=” is optional.

One option we have is to compose and define styles with other styles. If you have taken some form of an Object Oriented Programming course, we are employing Composition to avoid code repetition.

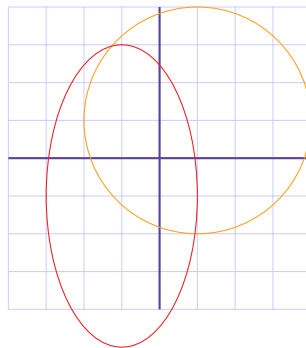


Figure 25: Modified Style

```

\tikzstyle Grid Line=[step=.5, color=white!80!blue, very thin]
\tikzstyle Axis Line=[thick, color=Violet]
\tikzstyle Circle Line=[color=YellowOrange]

\tikzstyle Ellipse Line=[Circle Line, color=Red]

\draw[style=Grid Line] (-2, -2) grid (2, 2);
\draw[style=Axis Line] (-2, 0) -- (2, 0);
\draw[Axis Line] (0, -2) -- (0, 2);
\draw[Circle Line] (0.5, 0.5) circle (1.5);
\draw[Ellipse Line] (-0.5, -0.5) ellipse (1 and 2);

```

4.2 Thickness

We have already seen “very thin” in latex, Here are examples of other thickness of paths we can give to the draw command. Note that “thin” gives us the default thickness.

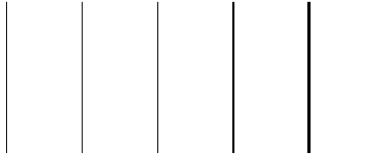


Figure 26: Thickness

```

\draw[ultra thin] (-2, 1) -- (-2, -1);
\draw[very thin] (-1, 1) -- (-1, -1);
\draw[thin] (0, 1) -- (0, -1);
\draw[thick] (1, 1) -- (1, -1);
\draw[very thick] (2, 1) -- (2, -1);
\draw[ultra thick] (3, 1) -- (3, -1);

```

4.3 Scaling



Figure 27: Small Circle

```

\tikz \draw (0,0) circle (0.5);

```

Suppose we want to enlarge the rendering without changing all the source code manually. We can do so with the environment option “scale”.

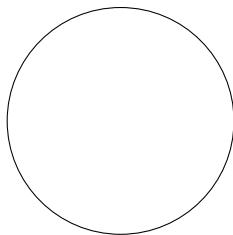


Figure 28: Scaled Circle

```
\begin{tikzpicture}[scale=3]
  \draw (0,0) circle (0.5);
\end{tikzpicture}
```

4.4 Scoping

We have previously been applying environment settings only to the “tikzpicture” environment, but what if we wanted for example to only change arrow head for a subset of the environment. This can be accomplished with scopes

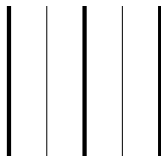


Figure 29: Basic Scoped Lines

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

4.5 Transformations

One powerful mechanism of TikZ are transformations

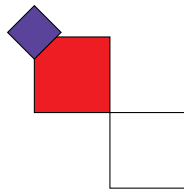


Figure 30: Basic Transformations

```
\filldraw[fill=Red] (0, 0) rectangle (1, 1);
\draw[xshift=1cm,yshift=-1cm] (0, 0) rectangle (1, 1);
\filldraw[fill=Violet,rotate=45] (0.5, 0.5) rectangle (1, 1);
```

4.6 For Loops

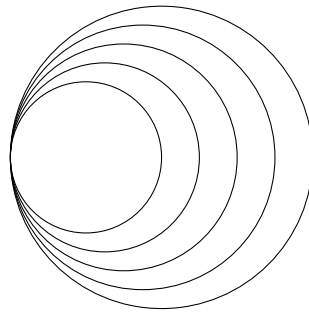


Figure 31: Basic Repetitions

```
\foreach \x in {1, 1.25, 1.5, 1.75, 2}
  \draw (\x, 0) circle (\x);
```

4.7 Layers

What if we wish to populate the background after drawing some nodes of shapes?

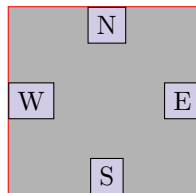


Figure 32: Basic Background Layer