An Introduction to Ti*k*Z

Wesley T. Honeycutt

University of Oklahoma

August 26, 2020

[0] https://github.com/BlueNalgene/Tikz_in_an_hour

Sources and Background

The Environment

Basic Tools

Libraries

Examples

Learning More

What Are We Doing Here?

- This workshop assumes you have some LATEXcompetency.

What Are We Doing Here?

- This workshop assumes you have some LaTeXcompetency.
- You can code along with a computer using your favorite editor. Overleaf will be used for examples.

What Are We Doing Here?

- This workshop assumes you have some LaTeXcompetency.
- You can code along with a computer using your favorite editor. Overleaf will be used for examples.
- This workshop covers concepts, full lists of tools are available online:
  - The Official Manual
  - A Very Minimal Iintroduction to Ti*k*Z
  - Overleaf's Ti*k*Z Manual
  - The Ti*k*Z Wikibook

Ti*k*Z Background

- Ti*k*Z is a language to control PGF (Portable Graphics Format).

Ti*k*Z Background

- Ti*k*Z is a language to control PGF (Portable Graphics Format).
- Ti*k*Z is an acronym for "Ti*k*Z ist *kein* Zeichenprogramm".

Ti*k*Z Background

- Ti*k*Z is a language to control PGF (Portable Graphics Format).
- Ti*k*Z is an acronym for "Ti*k*Z ist *kein* Zeichenprogramm".
- Ti*k*Z is interpreted by TEXderivative compilers and some graphics programs.

Ti*k*Z Background

- Ti*k*Z is a language to control PGF (Portable Graphics Format).
- Ti*k*Z is an acronym for "Ti*k*Z ist *kein* Zeichenprogramm".
- Ti*k*Z is interpreted by TEXderivative compilers and some graphics programs.
- Lazy people (like me) can export Ti*k*Z code directly from many programs (e.g. Inkscape, Blender, Python, Gnuplot, R)

Summon TikZ Environment

The Ti*k*Z framework is contained in the `tikz` package.

The minimum for this environment would be:

```
\documentclass{minimal}
\usepackage{tikz}
\begin{document}
  content
\end{document}
```

Summon TikZ Environment

The Ti*k*Z framework is contained in the `tikz` package.

Ti*k*Z is called using the `tikzpicture` environment in LaTeX

The minimum for this environment would be:

```
\documentclass{minimal}
\usepackage{tikz}
\begin{document}
  \begin{tikzpicture}
    tikz content
  \end{tikzpicture}
\end{document}
```

Controlling the Environment

The `tikzpicture` environment is controlled like other LaTeX frames.

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
  \begin{figure}[t]
    \centering
    \begin{tikzpicture}
      content here
    \end{tikzpicture}
    \caption{Info about picture}
    \label{fig:my_label}
  \end{figure}
\end{document}
```

Controlling the *Picture*

The tikzpicture size should be controlled directly.

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
  \begin{tikzpicture}[scale=3]
    content here
  \end{tikzpicture}
  \\~\\
  \begin{tikzpicture}[xscale=3, yscale=2]
    more content here
  \end{tikzpicture}
\end{document}
```

The Basics of The Basics

- Ti*k*Z is high level abstraction of vector art commands

The Basics of The Basics

- Ti*k*Z is high level abstraction of vector art commands
- Ti*k*Z is 2D

The Basics of The Basics

- Ti*k*Z is high level abstraction of vector art commands
- Ti*k*Z is 2D
- 2D Vector = points, curves, and simple operations

The Basics of The Basics

- Ti*k*Z is high level abstraction of vector art commands
- Ti*k*Z is 2D
- 2D Vector = points, curves, and simple operations
- Size matters, reference doesn't

The Basics of The Basics

- Ti*k*Z is high level abstraction of vector art commands
- Ti*k*Z is 2D
- 2D Vector = points, curves, and simple operations
- Size matters, reference doesn't
- Ti*k*Z is ridiculously powerful. The manual is 1000+ pages for a reason.

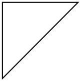
The General Case

```
\command[options, options, options] node connection;
```

- Within a Ti*k*Z picture environment, each part of a drawing gets a semicolon (;) terminated line.

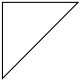
Arbitrary Reference 1

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

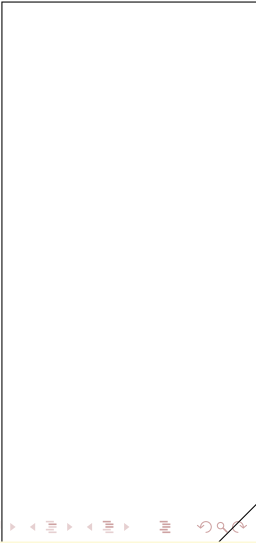


Arbitrary Reference 11

```
\begin{tikzpicture}
  \draw (10,10) -- (10,11) -- (11,11) -- cycle;
\end{tikzpicture}
```



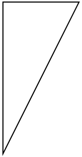
Size and Reference



```
\begin{tikzpicture}
  \draw (0,10) -- (0,10) -- (10,10) -- cycle;
\end{tikzpicture}
```

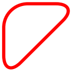
Default Unit = 1cm

```
\begin{tikzpicture}[x=1cm,y=2cm]
  \draw (0,0) -- (0,1) -- (1,1) -- cycle;
\end{tikzpicture}
```



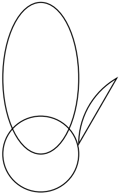
Drawings are Altered When Called by *draw*

```
\begin{tikzpicture}
  \draw[red, very thick, rounded corners=9pt] (0,0) -- (0,1)
     -- (1,1) -- cycle;
\end{tikzpicture}
```



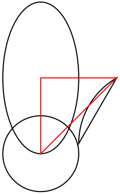
Altering Connections with Circles

```
\begin{tikzpicture}
  \draw (0,0) circle [radius=.5cm] (0,1) circle [x radius=.5cm, y radius=1cm] (1,1) arc (120:180:1) -- cycle;
\end{tikzpicture}
```



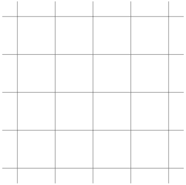
How did those Arcs work?

```
\begin{tikzpicture}
  \draw (0,0) circle [radius=.5cm] (0,1) circle [x radius=.5cm, y radius=1cm] (1,1) arc (120:180:1) -- cycle;
  \draw[red] (0,0) -- (0,1) -- (1,1) -- cycle;
\end{tikzpicture}
```



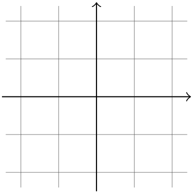
Grids with Defined Steps

```
\begin{tikzpicture}
  \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid
    (1.2,1.2);
\end{tikzpicture}
```



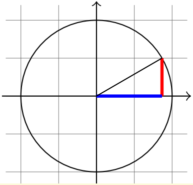
Coordinate Labels, Simple Arrows

```
\begin{tikzpicture}
  \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid
    (1.2,1.2);
  \draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
  \draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
\end{tikzpicture}
```



Right Angle Connections, Using Coordinate Labels

```
\begin{tikzpicture}
  \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid (1.2,1.2);
  \draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
  \draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
  \draw (0,0) circle (1cm);
  \draw[very thick,red] (30:1cm) --  (30:1cm |- x axis);
  \draw[very thick,blue] (30:1cm |- x axis) -- (0,0);
  \draw (0,0) -- (30:1cm);
\end{tikzpicture}
```



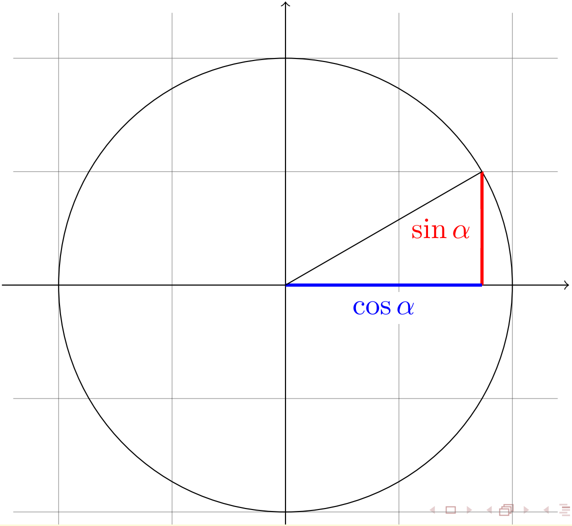
Nodes as Text Labels

Node Syntax:

node [*anchor, options*] {*contents*}

```
\begin{tikzpicture}[scale=3]
  \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid
    (1.2,1.2);
  \draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
  \draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
  \draw (0,0) circle (1cm);
  \draw[very thick,red] (30:1cm) -- node[left,fill=white]
    {$\sin \alpha$} (30:1cm |- x axis);
  \draw[very thick,blue] (30:1cm |- x axis) -- node[below=2
    pt,fill=white] {$\cos \alpha$} (0,0);
  \draw (0,0) -- (30:1cm);
\end{tikzpicture}
```

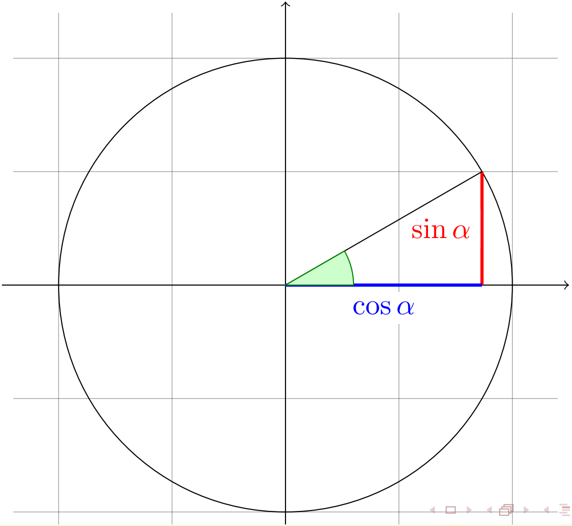
Nodes as Text Labels - Result



Custom Colors, Filled Areas

```
\begin{tikzpicture}[scale=3]
  \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid
    (1.2,1.2);
  \draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
  \draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
  \draw (0,0) circle (1cm);
  \draw[very thick,red] (30:1cm) -- node[left,fill=white]
    {$\sin \alpha$} (30:1cm |- x axis);
  \draw[very thick,blue] (30:1cm |- x axis) -- node[below=2
    pt,fill=white] {$\cos \alpha$} (0,0);
  \draw (0,0) -- (30:1cm);
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm
    ,0mm) arc (0:30:3mm) -- cycle;
\end{tikzpicture}
```

Custom Colors, Filled Areas - Result



Loops

```
\begin{tikzpicture}[scale=3]
  \draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid
    (1.2,1.2);
  \draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
  \draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
  \draw (0,0) circle (1cm);
  \draw[very thick,red] (30:1cm) -- node[left,fill=white]
    {$\sin \alpha$} (30:1cm |- x axis);
  \draw[very thick,blue] (30:1cm |- x axis) -- node[below=2
    pt,fill=white] {$\cos \alpha$} (0,0);
  \draw (0,0) -- (30:1cm);
  \filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm
    ,0mm) arc (0:30:3mm) -- cycle;
  \foreach \x/\xtext in {-1, -0.5/-\frac{1}{2}, 1}
  \draw (\x cm,1pt) -- (\x cm,-1pt) node[anchor=north,fill=
    white] {$\xtext$};
  \foreach \y/\ytext in {-1, -0.5/-\frac{1}{2}, 0.5/\frac
    {1}{2}, 1}
  \draw (1pt,\y cm) -- (-1pt,\y cm) node[anchor=east,fill=
```

Loop - Syntax

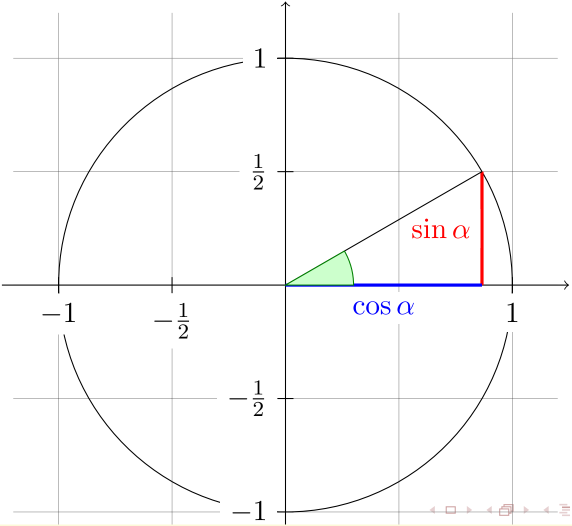
TikZ

content...

C

```
\begin{tikzpicture}[scale=3]
\draw[step=.5cm, gray, very thin] (-1.2,-1.2) grid
  (1.2,1.2);
\draw[->] (-1.25,0) -- (1.25,0) coordinate (x axis);
\draw[->] (0,-1.25) -- (0,1.25) coordinate (y axis);
\draw (0,0) circle (1cm);
\draw[very thick,red] (30:1cm) -- node[left,fill=white]
  {$\sin \alpha$} (30:1cm |- x axis);
\draw[very thick,blue] (30:1cm |- x axis) -- node[below=2
  pt,fill=white] {$\cos \alpha$} (0,0);
\draw (0,0) -- (30:1cm);
\filldraw[fill=green!20,draw=green!50!black] (0,0) -- (3mm
  ,0mm) arc (0:30:3mm) -- cycle;
```

Loops - Result



Libraries and How to Summon Them

Syntax in Preamble:

```
\usetikzlibrary{library}
```

- I can list all of the libraries with a minimal description, but there will not be time to give examples of each.
- I recommend this comprehensive Stack Overflow thread which attempts to provide an introduction for each library with examples.
- Official documentation in Part V of the manual.

Ti*k*Z Libraries in Brief (1/6)

How to read this list:

Descriptive Name (`Library command`) - One line description

Three Dimensions (`3D`) - Produce plots using cylindrical or spherical coordinate systems with predefined planes.

Angles (`angle`) - Draw angles between nodes and connections.

Arrow Tip (`arrows.meta`) - Add arrowheads and special dots to your node connections.

Automata (`automata`) - Draw finite automata and Turing machines.

Babel (`babel`) - Helps Ti*k*Z behave better with non-standard characters like ø.

TikZ Libraries in Brief (2/6)

Backgrounds (background) - Put a background behind your drawing.

Calculator (calc) - Uses TeXto calculate values.

Calendar calendar) - Draw a calendar.

Chains (chains) - Enable more complex connection between nodes.

Circuits (circuits) - Draw electronic circuits.

Decorations (decoration) - Fancy connections like squiggles, zig-zags, text, and shapes.

Entity-Relationship (er) - Tools for drawing entity-relationship coded diagrams.

Externalization (external) - Semi-automatic export of TikZ pictures.

Ti*k*Z Libraries in Brief (3/6)

Fading (`fadings`) - Create gradients between color and transparency.

Fitting (`fit`) - Fit a bounding box or circle around all the nodes you list in the command.

Fixed Points (`fixedpointarithmetic`) - Allow big numbers in calculations.

Floating Points (`fpu`) - Allow precise numbers in calculations.

Lindenmayer Systems (`lindenmayersystems`) - Draw branching and fractal designs.

Math (`math`) - Perform calculations in a user-friendly way.

Matrix (`matrix`) - Draw matrices and operations on them.

Mindmap (`mindmap`) - Draw *mindmap* style relationship trees.

Ti*k*Z Libraries in Brief (4/6)

Paper Folding `(folding)` - Draw objects which may be printed, cut, and then assembled into 3D objects.

Patterns `(patterns])` - Hatches, lines, dots, and other fill patterns.

Three Point Perspective `(perspective)` - Draw with up to 3 vanishing points for a 3D effect.

Petri-Net `(petri)` - Draw Petri-Net style logic diagrams

Plot Extension `(plothandlers)` - Adds even more ways you can use connections (partial lines, splines, gaps)

Plot Marks `(plotmarks)` - More shapes for your nodes.

Profiler `(profiler)` - Debugging tools and timers for compiling.

Ti*k*Z Libraries in Brief (5/6)

- Resource Description `(rdf)` - Output files with more descriptive comments to make them human-readable.
- Shading `(shadings)` - Creates color gradients.
- Shadow `(shadows)` - Create drop shadows behind nodes and connections.
- Shapes `(shapes)` - Add pre-defined common shapes.
    - Multipart `(shapes.multipart)` - Shapes with dividing lines.
    - Callouts `(shapes.callouts)` - Create callouts (speech bubbles).
    - Misc `(shapes.misc)` - More pre-defined shapes.
- Spy `(spy)` - Spy on or zoom in on part of your drawing like an inset map.

TikZ Libraries in Brief (6/6)

SVG Path (`svg.path`) - Create your own connection paths using SVG rules.

To Path (`topaths`) - Treat your connections as a “path” for vector outputs.

Through Points (`through`) - Make your connections go *through* a node rather than *to* a point.

Tree (`trees`) - Create complex tree connections.

Turtle Graphics (`turtle`) - Draw using “turtle graphics” commands rather than pre-defined nodes.

Views (`views`) - Define special rules for the box that contains a TikZ graphic.

Some Practical Examples

You can do nigh-infinite things with Ti*k*Z

Here are some examples which touch on useful concepts:

- MOSFET - Using simple nodes to create diagrams.
- Amplitude and Frequency
- and more...

MOSFET (1/5)

We can use the LaTeX definitions to define parts of our drawing in the preamble of our document.

Custom Colors

```
\newcommand{\metalone}{[pattern= horizontal lines, pattern
    color=blue]}
```

For this example, I have defined: `metalone`, `metaltwo`, `metalthree`, `poly`, `pdiff`, `ndiff`, `pwell`, `nwell`, `oxide`, and `silicon`.

MOSFET (2/5)

We can tell connections to make a curve

Angles In and Out

```
(1,2.5) to [out=270,in=180] (1.5,2)
```

MOSFET (3/5)

We can connect to a node at certain anchor points and add text.

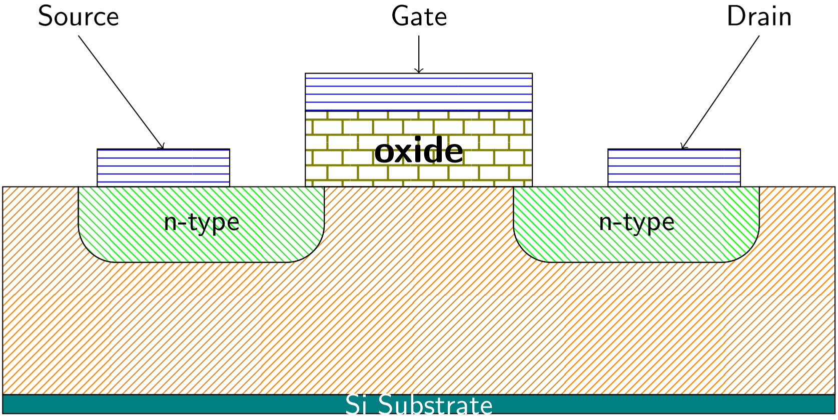
Anchors and Text

```
(0,.25) node [midway,above] {p doped Si}
```

MOSFET (4/5)

```
\begin{tikzpicture}
\draw \pdiff (0,.25) -- (0,3) -- (1,3) -- (1,2.5) to [out=270,in=180] (1.5,2) -- (3.75,2) to [out=0,in=270] (4.25,2.5) -- (4.25,3) -- (6.75,3) -- (6.75,2.5) to [out=270,in=180] (7.25,2) -- (9.5,2) to [out=0,in=270] (10,2.5) -- (10,3) -- (11,3) -- (11,.25) -- ;
\draw \metalthree (0,0) rectangle (11,.25) node [midway, color=white]
{Si Substrate};
\draw \oxide (4,3) rectangle (7,4) node [pos=.5,font=\bf\Large] {oxide};
\draw \metalone (4,4) rectangle (7,4.5);
\draw \ndiff (4.25,3) -- (1,3) -- (1,2.5) to [out=270,in=180] (1.5,2) -- (3.75,2) to [out=0,in=270] (4.25,2.5) -- (4.25,3) node at (2.625,2.5) [align=center] {n-type};
\draw \ndiff (10,3) -- (6.75,3) -- (6.75,2.5) to [out=270,in=180] (7.25,2) -- (9.5,2) to [out=0,in=270] (10,2.5) -- (10,3) node at (8.375,2.5) [align=center] {n-type};
\draw \metalone (1.25,3) rectangle (3,3.5);
\draw \metalone (8,3) rectangle (9.75,3.5);
\draw [->] (1,5) node [above] {Source} -- (2.125,3.5);
\draw [->] (10,5) node [above] {Drain} -- (8.975,3.5);
\draw [->] (5.5,5) node [above] {Gate} -- (5.5,4.5);
\node at (5.5,-.5) [align=center] {$V_{GS} < V_{threshold}$};
\end{tikzpicture}
```

MOSFET (5/5)



$$V_{GS} < V_{threshold}$$

Amplitude and Frequency (1/5)

Let's change course and render a plot to show how amplitude and period of a trigonometric function is altered:

$$f\left(x\right) = A * \left(\sin\left(B * \theta\right)\right)$$

We will use a new library:

```
\usetikzlibrary{datavisualization.formats.functions}
```

Amplitude and Frequency (2/5)

- We first call Data Visualization.
- We describe the appearance of the plot (axes, grids).
- We describe the lines (smooth, colors, dashes).
- We add legend entries for each plot.
- Finally, we tell it to expect functions.

```
\begin{tikzpicture}
\datavisualization [
school book axes,
y axis=grid,
x axis=grid,
visualize as smooth line/.list={sina,
     sinb,sinc},
style sheet=strong colors,
style sheet=vary dashing,
sina={label in legend={text=$\sin x$}},
sinb={label in legend={text=$ 3 \times{\
     sin x}$}},
sinc={label in legend={text=$\sin{\left
     (3\times x \right)}$}},
data/format=function]

```

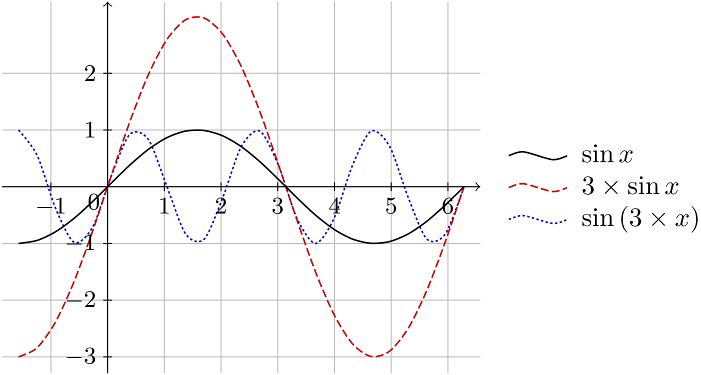
Amplitude and Frequency (3/5)

- Each function gets a `data` entry with a `set` label.
- Variables are defined in an `interval`.
- Functions are defined with `func`.
- Since we are using radians, we have to tell it `r`.

```
data [set=sina] {
var x : interval [-0.5*pi:2*pi];
func y = sin(\value x r);
}
data [set=sinb] {
var x : interval [-0.5*pi:2*pi];
func y = 3 * sin(\value x r);
}
data [set=sinc] {
var x : interval [-0.5*pi:2*pi];
func y = sin(3 * \value x r);
};
\end{tikzpicture}
```

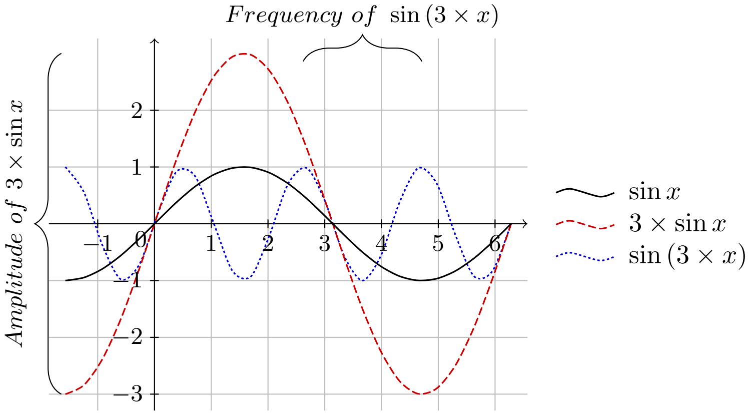
Amplitude and Frequency (4/5)

Our complete plot:



Amplitude and Frequency (5/5)

We can add nodes like before:



Where do I find more?

Is there something specific you want to see as an example?
https://texample.net/tikz/examples/

RTFM

RTFM

When in doubt: Google

Ti*k*Z has rolled into the TEXcommunity.

When in doubt: Google

TikZ has rolled into the TEXcommunity.

The community loves to help.

When in doubt: Google

Ti*k*Z has rolled into the TEXcommunity.

The community loves to help.

https://tex.stackexchange.com/

At your local library

OU libraries has a LaTeX expert:

Amanda Schilling
🔗: Stem Services
☎: (405) 325-6126
✉: amanda.schilling@ou.edu

Office Hours: W/Th 8-9am in DAVIS
M 6-8pm in the Learning Lab

At your local library

OU libraries has a LaTeX expert:

Mark Laufersweiler
🔗: Research Data Specialist
☎: (405) 325-3710
✉: laufers@ou.edu

Or contact me

I'm just a LaTeX junkie:

Wesley T. Honeycutt
🔗: Personal Site
📞: *I have an office phone?*
✉: honeycutt@ou.edu
: https://github.com/BlueNalgene