

1 基本的なコマンド

- `\draw = \path[draw]`
- `\fill = \path[fill]`
- `\filldraw = \path[fill,draw] or \path [fill] [draw]`
- `\pattern = \path[pattern]`
- `\shade = \path[shade]`
- `\shadedraw = \path[shade,draw]`
- `\clip = \path[clip]`
- `\useasboundingbox = \path[use as bounding box]`

2 color オプション

- `color=color1!number!color2`

`color1:color2=number:(100-number)` の割合で色を塗る. `!color2` を省略すると `white` と見做される. `!number!color2` を省略すると `color1` のみで色を塗る.



```
\begin{tikzpicture}
\draw [color=red!50!blue] (0,0)--(2,0);
\draw [fill,color=yellow!30] (0,.5) circle (.5cm);
\draw [fill,color=green] (1,.5) rectangle (1.5,1);
\end{tikzpicture}
```

- `\colorlet{color name}{color def}` で自由に色を定義できる.



```
\colorlet{hoge hoge}{blue!25!yellow}
\tikz \draw[fill,color=hoge hoge] (0,0) circle (1em);
```

3 関数の描画

- `plot[local option] coordinate {(coord1)(coord2)...(coordn)}`

この指定方法は `\draw[local option] (coord1)--(coord2)--...--(coordn)` と同じである.



```
\tikz \draw plot coordinates {(0,0)(1,0)(1,1)(-30:2)};
```

- `plot[local option] file {file name}`

これはすでに作成済みの `.table` ファイルから座標を読み込むものである. 必要ならばマニュアルの 224 ページを参照すること.

- `plot[local option] (coordinate expression)`

関数式の指定で任意の関数を描画する方法.

– `variable=macro`

関数の変数を指定する. default では `\x` である.

– `samples=number`

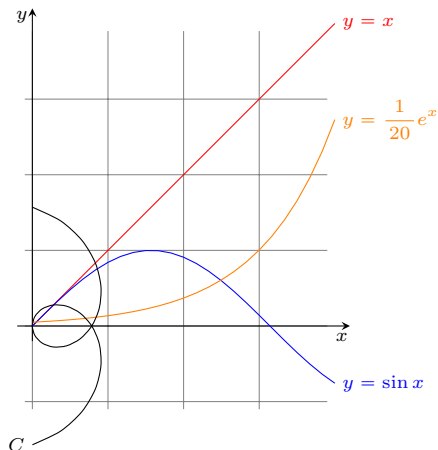
描画に使用する制御点の個数を指定する. default では 25 である.

– `domain=start:end`

関数の定義域を指定する. default では `start=-5`, `end=5` である.

– `samples at={sample list}`

制御点のうちどれを使用するかを指定する.



```
\begin{tikzpicture}[domain=0:4,>=stealth]
\scriptsize
\draw[very thin,color=gray](-0.1,-1.1)grid(3.9,3.9);

\draw[->](-.2,0)--(4.2,0)node[xshift=-3pt,yshift=-4pt]{$x$};
\draw[->](0,-.2)--(0,4.2)node[xshift=-4pt,yshift=-3pt]{$y$};

\draw[color=red]plot(\x,\x)node[right]{$y=x$};
\draw[color=blue]plot(\x,{sin(\x r)})node[right]{$y=\sin x$};
\draw[color=orange]plot(\x,{exp(\x)*1/20})node[right]{$y=\frac{1}{20}e^x$};

\draw[scale=0.5,domain=-3.1415:3.1415,smooth,variable=\t]
plot({\t*sin(\t r)},{\t*cos(\t r)})node[left]{$C$};
\end{tikzpicture}
```

• `plot[local option] function {gunplot formula}`

この指定方法は--`shell-escape` オプションが必要になり, `\jobname.idname.gnuplot` および `\jobname.idname.table` という補助ファイルが作られる.

– `parametric=true or false`

媒介変数を使用するかどうかを指定する. default で `true` であり, `parametric` とすると自動で `true` が適用される.

– `id=idname`

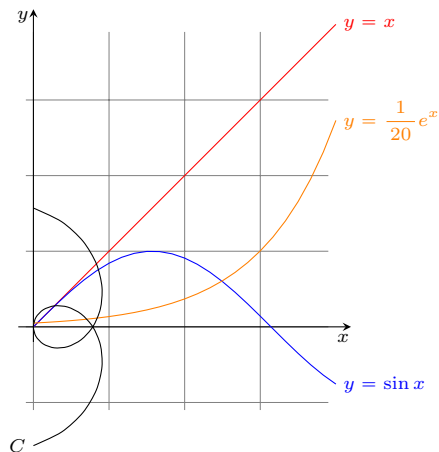
補助ファイルは基本的に `\jobname.idname.gnuplot`, `\jobname.idname.table` という書式である

が、この `idname` の部分を指定する。通常は一意の名称を使用する。

– row `gnuplot`

– `evry plot`

すべてのプロットで使用するスタイルなどを指定する。例えば、補助ファイルをすべて `./plots` ディレクトリに作成する場合は `ex:\tikzset{every plot/.style={prefix=plots/}}` と設定する。



```
\begin{tikzpicture}[domain=0:4,>=stealth]
\scriptsize
\draw[very thin,color=gray](-0.1,-1.1)grid(3.9,3.9);

\draw[->](-.2,0)--(4.2,0)node[xshift=-3pt,yshift=-4pt]{$x$};
\draw[->](0,-.2)--(0,4.2)node[xshift=-4pt,yshift=-3pt]{$y$};

\draw[color=red]plot[id=tikz-sample-function-x]
function{x}node[right]{$y=x$};
\draw[color=blue]plot[id=tikz-sample-function-sin]
function{sin(x)}node[right]{$y=\sin x$};
\draw[color=orange]plot[id=tikz-sample-function-exp]
function{exp(x)*1/20}node[right]{$y=\frac{1}{20}e^x$};

\draw[scale=0.5,domain=-3.1415:3.1415,smooth,parametric,%
id=tikz-sample-function-param]
plot function{t*sin(t),t*cos(t)}node[left]{$C$};
\end{tikzpicture}
```

4 線の描画

4.1 線の太さの数値による指定

- `line width=dimension`

`dimension` に単位付き数値で指定する。default は 0.4pt (thin) である。

```
\tikz \draw[line width=10pt] (0,0)--(1,0);
```

4.2 線の太さの名前による指定

- `ultra thin`: 線に太さを 0.1pt に設定する.
`—— \tikz \draw[ultra thin] (0,0)--(1,0);`
- `very thin`: 線に太さを 0.2pt に設定する.
`—— \tikz \draw[very thin] (0,0)--(1,0);`
- `thin`: 線に太さを 0.4pt に設定する.
`—— \tikz \draw[thin] (0,0)--(1,0);`
- `semithick`: 線の太さを 0.6pt に設定する.
`—— \tikz \draw[semithick] (0,0)--(1,0);`
- `thick`: 線の太さを 0.8pt に設定する.
`—— \tikz \draw[thick] (0,0)--(1,0);`
- `very thick`: 線の太さを 1.2pt に設定する.
`—— \tikz \draw[very thick] (0,0)--(1,0);`
- `ultra thick`: 線の太さを 1.6pt に設定する.
`—— \tikz \draw[ultra thick] (0,0)--(1,0);`

4.3 線の端末の処理

- `line cap=type`

`type` には, “round”, “rect”, “butt” の 3 種類があり, default は `butt` であり両端点は与えられた始点および終点の場所で終わる. `rect` は `butt` の両端に線の太さの半分だけ長方形が付け加えられる. `round` は `butt` の両端に線の太さの半分の半円が付け加えられる.



```
\begin{tikzpicture}
\draw[color=green] (0,-.5)--(0,1.5) (1,-.5)--(1,1.5);
\begin{scope}[line width=10pt]
\draw[line cap=round] (0,0)--(1,0);
\draw[line cap=rect] (0,.5)--(1,0.5);
\draw[line cap=butt] (0,1)--(1,1);
\end{scope}
\draw[color=white,line width=1pt] (0,0)--(1,0) (0,.5)--(1,.5) (0,1)--(1,1);
\end{tikzpicture}
```

4.4 線分の繋ぎ方

- `line join=type`

`type` には, “round”, “bevel”, “miter” の 3 種類あり, default は “miter” である.



```
\begin{tikzpicture}[line width=10pt]
  \draw[line join=round] (0,0)---+ (.5,1)---+ (.5,-1);
  \draw[line join=bevel] (1.5,0)---+ (.5,1)---+ (.5,-1);
  \draw[line join=miter] (3,0)---+ (.5,1)---+ (.5,-1);
\end{tikzpicture}
```

また, `default(miter)` かつ非常に鋭角なコーナーのとき, 折れ点から伸ばす部分が線の太さの 10 倍になると自動で `bevel` になるようにしてある. これを変更するには `miter limit` オプションで変更する. `default` では 10 が設定してある.



```
\begin{tikzpicture}[line width=5pt]
  \draw (0,0)---+ (5,.5)---+ (-5,.5);
  \draw[miter limit=25] (6,0)---+ (5,.5)---+ (-5,.5);
\end{tikzpicture}
```

4.5 破線の数値による指定

- `dash pattern=pattern`

`pattern` に具体的に破線の使用を記述する.

```
\tikz \draw[dash pattern=on 2pt off 3pt on 4pt off 4pt] (0pt,0pt)--(3.5cm,0pt);
```

- `dash phase=dimension`

破線の開始を `dimension` の分だけずらす. `default` では 0pt.

==_==_==_==_==

```
\begin{tikzpicture}[dash pattern=on 20pt off 10pt]
  \draw[dash phase=0pt] (0pt,3pt)--(3.5cm,3pt);
  \draw[dash phase=10pt] (0pt,0pt)--(3.5cm,0pt);
\end{tikzpicture}
```

4.5.1 破線・点線の名前による指定

- `solid`

_____ \tikz \draw[solid] (0,0)--(2,0);

- `dotted`

..... \tikz \draw[dotted] (0,0)--(2,0);

- `densely dotted`

```

..... \tikz \draw[densely dotted] (0,0)--(2,0);
• loosely dotted
..... \tikz \draw[loosely dotted] (0,0)--(2,0);
• dashed
----- \tikz \draw[dashed] (0,0)--(2,0);
• densely dashed
----- \tikz \draw[densely dashed] (0,0)--(2,0);
• loosely dashed
----- \tikz \draw[loosely dashed] (0,0)--(2,0);
• dashdotted
----- \tikz \draw[dashdotted] (0,0)--(2,0);
• densely dashdotted
----- \tikz \draw[densely dashdotted] (0,0)--(2,0);
• loosely dashdotted
----- \tikz \draw[loosely dashdotted] (0,0)--(2,0);
• dashdotdotted
----- \tikz \draw[dashdotdotted] (0,0)--(2,0);
• densely dashdotdotted
----- \tikz \draw[densely dashdotdotted] (0,0)--(2,0);
• loosely dashdotdotted
----- \tikz \draw[loosely dashdotdotted] (0,0)--(2,0);

```

4.6 矢線の設定

- `arrows=<start arrow kind>-<end arrow kind>`
 -の前後にそれぞれ矢線の始点・終点の形状を指定する。 `\usetikzlibrary` で `arrows` を読み込んでおく
 と様々な矢線の形状を利用できるようになる。


```

- <-----> \tikz \draw[to-to] (0,0)--(1,0);
- >-----< \tikz \draw[to reversed-to reversed] (0,0)--(1,0);
- ----- \tikz \draw[implies-implies] (0,0)--(1,0);
- <=====> \tikz \draw[double,implies-implies] (0,0)--(1,0);
- <-----> \tikz \draw[<->] (0,0)--(1,0);
- >-----< \tikz \draw[>-<] (0,0)--(1,0);
- <=====> \tikz \draw[<<->>] (0,0)--(1,0);
- >=====< \tikz \draw[>>-<<] (0,0)--(1,0);
- <=====> \tikz \draw[|<->|] (0,0)--(1,0);
- <-----> \tikz \draw[latex-latex] (0,0)--(1,0);
- >-----< \tikz \draw[latex reversed-latex reversed] (0,0)--(1,0);
- <-----> \tikz \draw[stealth-stealth] (0,0)--(1,0);
- >-----< \tikz \draw[stealth reversed-stealth reversed] (0,0)--(1,0);
- ○-----○ \tikz \draw[o-o] (0,0)--(1,0);
- ●-----● \tikz \draw[*-*] (0,0)--(1,0);
- ◆-----◆ \tikz \draw[diamond-diamond] (0,0)--(1,0);
- ◇-----◇ \tikz \draw[open diamond-open diamond] (0,0)--(1,0);

```

```

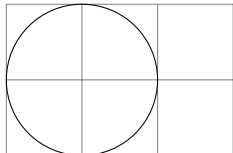
- ■——■ \tikz \draw[square-square] (0,0)--(1,0);
- □——□ \tikz \draw[open square-open square] (0,0)--(1,0);

```

5 円・弧の描画

- `coordinate circle [radius=dimension]`

`coordinate` を中心として半径 `dimension` の円を描く.



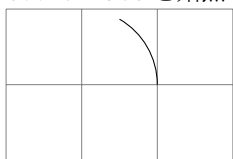
```

\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\draw (1,1) circle [radius=1cm];
\end{tikzpicture}

```

- `coordinate arc [start angle=degree1,end angle=degree2,radius=dimension]`

`coordinate` を始点とした半径 `dimension`, 開始角 `degree1`, 終了角 `degree2` の弧を描く.



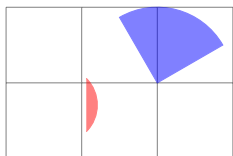
```

\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\draw (2,1) arc [start angle=0,end angle=60,radius=1cm];
\end{tikzpicture}

```

`start angle` または `end angle` のどちらか一方と `delta angle=degree` として中心角を指定する方法もある。また, `radius` の代わりに `x radius`, `y radius` を使用すると楕円の弧となる。

ただし, このままで非常に使いにくいから `\mytikzarc[option](中心){半径}{開始角}{終了角}` というコマンドを自作した。コマンドのすぐ後ろに*をつけると弧と中心を結んだ扇形を描く。 `option` は `\draw` のオプションである。



```

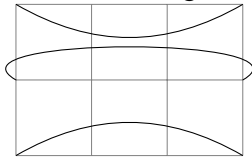
\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\mytikzarc*[fill,blue,opacity=.5](2,1){1cm}{30}{120}
\mytikzarc[fill,red,opacity=.5](45:1){.5cm}{-45}{45}
\end{tikzpicture}

```

sector を省略して fill を指定すると扇形ではなく弓形の塗りつぶしとなる。

- `\draw coordinate1 to [out=degree1,in=degree2] coordinate2`

2 点 `coordinate1`, `coordinate2` を結ぶ線分と角度 `degree1` で点 `coordinate1` から出発して点 `coordinate2` へ角度 `degree2` で到着する滑らかな曲線を描く。



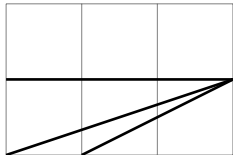
```
\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\draw(0,0) to [out=30,in=150] (3,0);
\draw(3,1) to [out=30,in=150] (0,1);
\draw(3,2) to [out=-150,in=-30] (0,2);
\end{tikzpicture}
```

6 点および交点の設定

6.1 点の設定

- `\coordinate (点の名前) at (座標)`

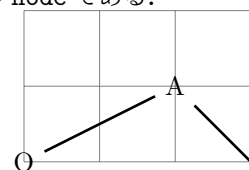
`\path coordinate` の短縮形。 `\draw` など線で引いている途中でも (座標) `coordinate` (点の名前) で点の名前を設定できる。



```
\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\coordinate (O) at (0,0);
\draw(0,1)--(3,1) coordinate (A)--(1,0);
\draw(O)--(A);
\end{tikzpicture}
```

- `\node [オプション] (点の名前) at (座標) {ラベル}`

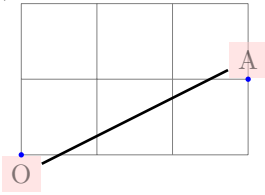
`\path node` の短縮形。 `\draw` など線で引いている途中でも (座標) `node [オプション]` (点の名前) {ラベル} で点の名前などを設定できる。 `node` には形・大きさが存在する。 `coordinate` は大きさが 0 の `node` である。



– 位置

* above, below, right, left

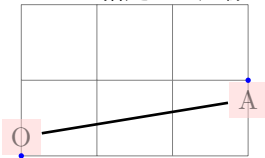
それぞれ指定した座標の上, 下, 右, 左に node を設定する. 座標に名前をつけてその上, 下, 右, 左にラベルを表示するのではないことに注意.



```
\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\node[fill,red!20,opacity=.5,text=black](O)at(0,0)[below]{O};
\node[fill,red!20,opacity=.5,text=black](A)at(3,1)[above]{A};
\draw(O)--(A);
\fill[blue](O.north)circle[radius=1pt](A.south)circle[radius=1pt];
\end{tikzpicture}
```

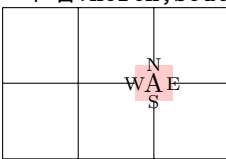
* anchor=north,south,east,west

それぞれ指定した座標に node の北, 南, 東, 西が重なるように node を設定する.



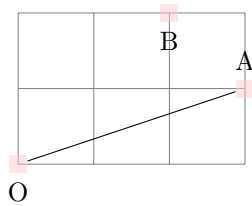
```
\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\node[fill,red!20,opacity=.5,text=black](O)at(0,0)[anchor=south]{O};
\node[fill,red!20,opacity=.5,text=black](A)at(3,1)[anchor=north]{A};
\draw(O)--(A);
\fill[blue](O.south)circle[radius=1pt](A.north)circle[radius=1pt];
\end{tikzpicture}
```

* ノード名.north,south,east,west



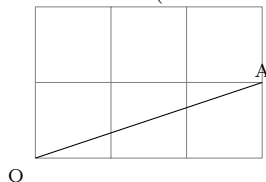
```
\begin{tikzpicture}
\draw(0,0)grid(3,2);
\node[fill,red!20,text=black](A)at(2,1){A};
\scriptsize
\path node at(A.north){N} node at(A.south){S};
\node at(A.west){W};
\node at(A.east){E};
\end{tikzpicture}
```

— ラベルオプションで label=位置: ラベルと指定してラベルの位置をずらす.



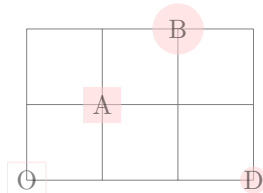
```
\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\node[fill,red!20,opacity=.5,text=black](O)at(0,0)[label=south:O]{};
\node[fill,red!20,opacity=.5,text=black](A)at(3,1)[label=north:A]{};
\node[fill,red!20,opacity=.5,text=black](B)at(2,2)[label=below:B]{};
\draw(O)--(A);
\end{tikzpicture}
```

ただし、node には大きさがあるので上記のように 2 つの node を線で結ぶと node の近辺は描かれな
い。そこで下記のように coordinate を使ってラベルと配置したほうが良いだろう。inner sep を設
定することによって点とラベルとの距離も任意に設定できる。ただし、label=num:label と指定する
こともできるが (num は角度)、この場合は inner sep での指定は効果が無いようである。



```
\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\coordinate (O) at (0,0);
\coordinate (A) at (3,1);
\draw (O)--(A);
\node[label=-135:{\scriptsize O},inner sep=0pt] at (O){};
\node[anchor=south,inner sep=2pt] at (A){\scriptsize A};
\end{tikzpicture}
```

- 形状 circle, rectangle, coordinate の 3 種類が設定可能。デフォルトは rectangle であり、ライ
ブラリで拡張可能である。オプションに draw をつけると輪郭が描かれ、fill をつけると内部が塗りつ
ぶされる。また、inner sep=dimension で node 内のラベルと外枠との距離を指定できる。デフォル
トは .3333em である。



```
\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\node[draw,red!20,opacity=.5,text=black](O)at(0,0){O};
\node[fill,red!20,opacity=.5,text=black](A)at(1,1){A};
\end{tikzpicture}
```

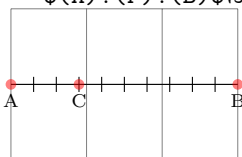
```

\node[fill,circle,red!20,opacity=.5,text=black](B)at(2,2){B};
\node[fill,circle,red!20,opacity=.5,text=black](C)at(3,1){C};
\node[fill,circle,inner sep=0pt,red!20,opacity=.5,text=black](D)at(3,0){D};
\end{tikzpicture}

```

• 内分点・外分点など

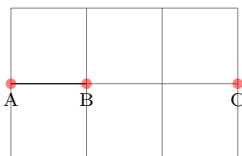
- $(A)!\text{num}!(B)$ で線分 AB を $\text{num} : (1 - \text{num})$ に内分 ($0 < \text{num} < 1$ なら内分点, $\text{num} < 0$ または $1 < \text{num}$ ならば外分点) する点を表す. 別の言い方をすれば, $\text{num} \times \overrightarrow{AB}$ の端点を表す.
- $(A)!\text{num}!\text{angle}:(B)$ では \overrightarrow{AB} を num 倍したものを点 A を中心として angle 回転させた端点を表す.
- $(A)!\text{dim}!(B)$ とすると線分 AX の長さが dim となる直線 AB 上の点 X を表す. これは $(A)!\text{num}!\text{angle}:(B)$ と同様に回転させることもできる.
- $(A)!(P)!(B)$ は点 P から直線 AB に下ろした垂線の足の座標を表す.



```

\begin{tikzpicture}
\draw[help lines] (0,-1)grid(3,1);
\coordinate (A) at (0,0);
\coordinate (B) at (3,0);
\coordinate (C) at ($(A)!.3!(B)$);
\draw(A)--(B);
\foreach \num in {.1,.2,...,.9}
\draw($(A)!\num!(B)+(0,-.1)$)--($(A)!\num!(B)+(0,.1)$);
\foreach \point in {A,B,C}{
\fill[red,opacity=.5] (\point) circle [radius=2pt];
\node at(\point)[below]{\scriptsize\point};
}
\end{tikzpicture}

```



```

\begin{tikzpicture}
\draw[help lines] (0,-1)grid(3,1);
\coordinate (A) at (0,0);
\coordinate (B) at (1,0);
\coordinate (C) at ($(A)!3!(B)$);
\draw(A)--(B);
\foreach \point in {A,B,C}{
\fill[red,opacity=.5] (\point) circle [radius=2pt];
\node at(\point)[below]{\scriptsize\point};
}
\end{tikzpicture}

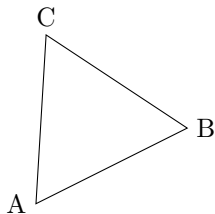
```

`\end{tikzpicture}`



```
\begin{tikzpicture}[>=stealth,line width=1pt]
\draw[help lines](0,0)grid(3,2);
\coordinate[label=left:A](A)at(0,1);
\coordinate[label=left:B](B)at(0,0);
\coordinate[label=right:C](C)at($(A)!3!90:(B)$);
\foreach \point in {A,B,C}{
\fill[red,opacity=.5](\point)circle[radius=2pt];
}
\draw[->](A)--(B);
\draw[->](A)--(C);
\end{tikzpicture}
```

さらに、次のように繰り返し使うこともできる.

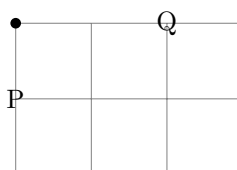


```
\begin{tikzpicture}
\coordinate[label=left:A](A)at(0,0);
\coordinate[label=right:B](B)at(2,1);
\coordinate[label=above:C](C)at($(A)!.5!(B)!\sin(60)*2!90:(B)$);
\draw(A)--(B)--(C)--cycle;
\end{tikzpicture}
```

6.2 交点の設定

- 直交する2直線の交点

((Pの座標)|(Qの座標)) Pを通る垂直線とQを通る水平線との交点.

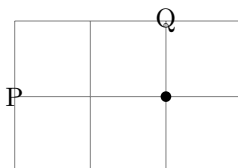


```

\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\coordinate(P)at(0,1);
\coordinate(Q)at(2,2);
\node at (P){P};
\node at (Q){Q};
\fill(P|-Q) circle [radius=2pt];
\end{tikzpicture}

```

((P の座標)-|(Q の座標)) P を通る水平線と Q を通る垂直線との交点.



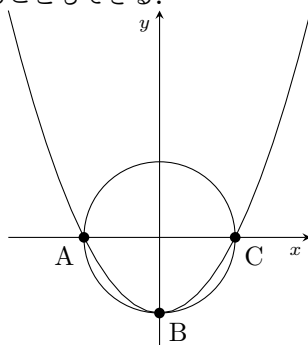
```

\begin{tikzpicture}
\draw[gray,very thin](0,0)grid(3,2);
\coordinate(P)at(0,1);
\coordinate(Q)at(2,2);
\node at (P){P};
\node at (Q){Q};
\fill(P-|Q) circle [radius=2pt];
\end{tikzpicture}

```

- 任意のパスの交点

異なる 2 つのパスに `name path=path1` と `name path=path2` を設定し, `name intersections={of=path1 and path2}` で交点を設定する. 交点の名前はデフォルトでは `intersection-1`, `intersection-2`, ... とつけられるが, `name intersections` の引数に `by={A,B,C,...}` と指定することもできる.



```

\begin{tikzpicture}[>=stealth]
\draw[->](-2,0)--(2,0) node [below left]{\scriptsize$x$};
\draw[->](0,-1.5)--(0,3) node [below left]{\scriptsize$y$};
\draw[name path=para] plot [domain=-2:2] (\x,{pow(\x,2)-1});
\draw[name path=circ] (0,0) circle [radius=1];

```

```

\path [name intersections={of=para and circ,by={A,B,C}}];
\fill (A) circle [radius=2pt] (B) circle [radius=2pt] (C) circle [radius=2pt];
\node at (A)[below left]{A};
\node at (B)[below right]{B};
\node at (C)[below right]{C};
\end{tikzpicture}

```

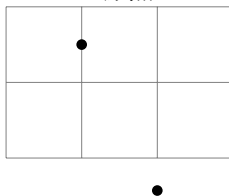
7 Coordinate Systems

7.1 Canvas, XYZ, and Polar Coordinate Systems

7.1.1 Coordinate system canvas

- `canvas cs:x`, `canvas cs:y`

それぞれ原点からの x 軸, y 軸方向への距離を指定する.

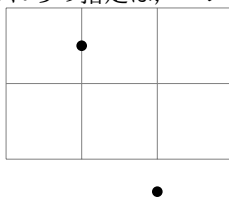


```

\begin{tikzpicture}
\draw[help lines] (0,0)grid(3,2);
\fill(canvas cs:x=1cm,y=1.5cm)circle[radius=2pt];
\fill(canvas cs:x=2cm,y=-5mm+2pt)circle[radius=2pt];
\end{tikzpicture}

```

これらの指定は, コンマで区切られた 2 つの dimension を使って簡単に記述することができる.



```

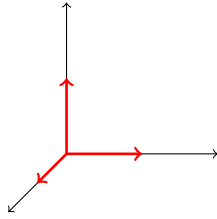
\begin{tikzpicture}
\draw[help lines] (0,0)grid(3,2);
\fill(1cm,1.5cm)circle[radius=2pt];
\fill(2cm,-5mm+2pt)circle[radius=2pt];
\end{tikzpicture}

```

7.1.2 Coordinate system xyz

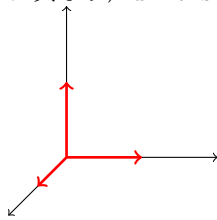
- `xyz cs:x`, `xyz cs:y`, `xyz cs:z`

それぞれ x 軸, y 軸, z 軸の単位ベクトルを何倍するかを指定する.



```
\begin{tikzpicture}[->]
  \draw(0,0)--(xyz cs:x=2);
  \draw(0,0)--(xyz cs:y=2);
  \draw(0,0)--(xyz cs:z=2);
  \begin{scope}[red,line width=1pt]
    \draw(0,0)--(xyz cs:x=1);
    \draw(0,0)--(xyz cs:y=1);
    \draw(0,0)--(xyz cs:z=1);
  \end{scope}
\end{tikzpicture}
```

これも、カンマで区切られた3つの数値によって簡単に記述することができるが、canvas coordinate systemとは異なり、dimensionを記述することはできない。



```
\begin{tikzpicture}[->]
  \draw(0,0)--(2,0);
  \draw(0,0)--(0,2,0);
  \draw(0,0)--(0,0,2);
  \begin{scope}[red,line width=1pt]
    \draw(0,0)--(1,0);
    \draw(0,0)--(0,1,0);
    \draw(0,0)--(0,0,1);
  \end{scope}
\end{tikzpicture}
```

7.1.3 Coordinate system canvas polar

- canvas polar cs:angle, canvas polar cs:radius

それぞれ極座標の偏角 (60 分法) と動径を指定する。

／ `\tikz\draw(0,0)--(canvas polar cs:angle=30,radius=1cm);`

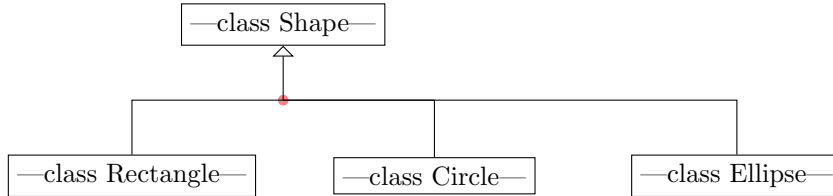
これはコロンで区切られた数値と dimension によって簡単に記述することができる。

／ `\tikz\draw(0,0)--(30:1cm);`

7.2 Node Coordinate System

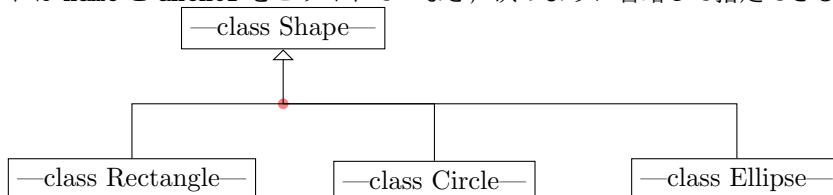
7.2.1 Coordinate system node

- `node cs:name`
予め設定されたノードへアクセスする.
- `node cs:anchor`
ノードのアンカーを指定する.



```
\begin{tikzpicture}
  \node(shape)at(0,2)[draw]{|class Shape|};
  \node(rect)at(-2,0)[draw]{|class Rectangle|};
  \node(circle)at(2,0)[draw]{|class Circle|};
  \node(ellipse)at(6,0)[draw]{|class Ellipse|};
  \fill[red!50](0,1)circle[radius=2pt];
  \draw(node cs:name=circle,anchor=north)|-(0,1);
  \draw(node cs:name=ellipse,anchor=north)|-(0,1);
  \draw[-open triangle 90](node cs:name=rect,anchor=north)
    |-(0,1)-|(node cs:name=shape,anchor=south);
\end{tikzpicture}
```

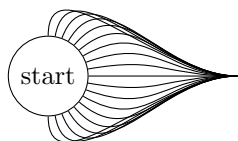
これは `name` と `anchor` をピリオドでつなぎ、次のように省略して指定できる.



```
\begin{tikzpicture}
  \node(shape)at(0,2)[draw]{|class Shape|};
  \node(rect)at(-2,0)[draw]{|class Rectangle|};
  \node(circle)at(2,0)[draw]{|class Circle|};
  \node(ellipse)at(6,0)[draw]{|class Ellipse|};
  \fill[red!50](0,1)circle[radius=2pt];
  \draw(circle.north)|-(0,1);
  \draw(ellipse.north)|-(0,1);
  \draw[-open triangle 90](rect.north)
    |-(0,1)-|(shape.south);
\end{tikzpicture}
```

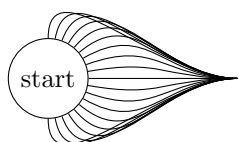

- `node cs:angle`

`anchor` を使ってキーワードで指定する代わりに角度 (60 分法) で指定する.



```
\begin{tikzpicture}
  \node(start)[draw,shape=circle]{start};
  \foreach \angle in {-90,-80,...,90}
    \draw(node cs:name=start,angle=\angle)
      .. controls +(\angle:1cm) and +(-1,0) .. (2.5,0);
\end{tikzpicture}
```

これも `name` と `angle` をピリオドでつなぎ, 次のように省略して指定できる.

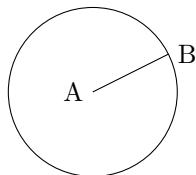


```
\begin{tikzpicture}
  \node(start)[draw,shape=circle]{start};
  \foreach \angle in {-90,-80,...,90}
    \draw(start.\angle)
      .. controls +(\angle:1cm) and +(-1,0) .. (2.5,0);
\end{tikzpicture}
```

8 座標・数値の保存

- `\p<digit>`, `\x<digit>`, `\y<digit>`, `\n<digit>`

`\p1` に (3, 5) が設定されたとき, `\x1=3`, `\y1=5` が設定される. これを利用すると, 指定した 2 点のうち一方を中心として他方を通る円の作図などができる. `\n<digit>` には任意の数値を代入できる.

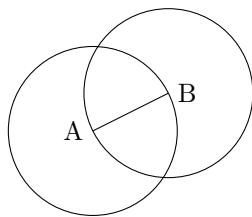


```
\begin{tikzpicture}
  \coordinate[label=left:A](A) at (1,1);
  \coordinate[label=right:B](B) at (3,2);
```

```

\draw(A)--(B);
\draw(A) let \p1=($(B)-(A)$),\n1={veclen(\x1,\y1)} in circle [radius=\n1];
\end{tikzpicture}

```

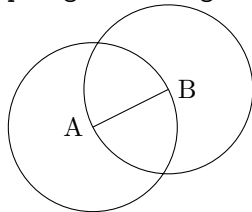


```

\begin{tikzpicture}
\coordinate[label=left:A](A) at (1,1);
\coordinate[label=right:B](B) at (3,2);
\draw(A)--(B);
\draw let \p1=($(B)-(A)$),\n1={veclen(\x1,\y1)}
in (A) circle [radius=\n1]
(B) circle [radius=\n1];
\end{tikzpicture}

```

$p<digit>$, $x<digit>$, $y<digit>$, $n<digit>$ の $<digit>$ は $\{name\}$ のように置き換えることができる.

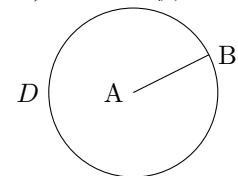


```

\begin{tikzpicture}
\coordinate[label=left:A](A) at (1,1);
\coordinate[label=right:B](B) at (3,2);
\draw(A)--(B);
\draw let \p{dis}=($(B)-(A)$),\n{r}={veclen(\x{dis},\y{dis})}
in (A) circle [radius=\n{r}]
(B) circle [radius=\n{r}];
\end{tikzpicture}

```

ただ、これらの例は `through` ライブラリを利用すれば次のように簡単に描画できる.



```

\begin{tikzpicture}[scale=.5]
\coordinate[label=left:A](A) at (1,1);

```

```

\coordinate[label=right:B] (B) at (3,2);
\draw(A)--(B);
\node[draw,circle through=(B),label=left:$D$] at (A){};
\end{tikzpicture}

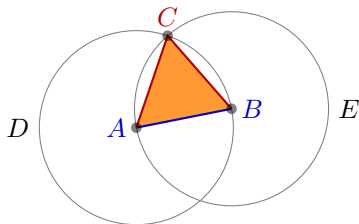
```

以上のことを駆使すると次のような描画ができる.

Proposition I

To construct an *equilateral triangle* on a given *finite straight line*.

Let *AB* be the given *finite straight line*. ...



```

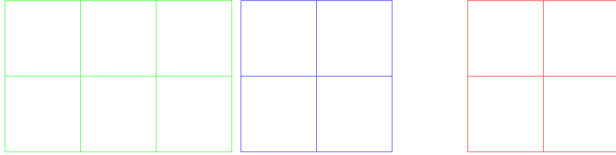
\pgfdeclarelayer{background}
\pgfsetlayers{background,main}
\begin{tikzpicture}[thick,help lines/.style={thin,draw=black!50}]
\def\A{\textcolor{input}{$A$}}\def\B{\textcolor{input}{$B$}}
\def\C{\textcolor{output}{$C$}}\def\D{$D$}\def\E{$E$}
\colorlet{input}{blue!80!black}
\colorlet{triangle}{orange}
\colorlet{output}{red!70!black}
\coordinate [label=left:\A] (A) at ($ (0,0) + .1*(rand,rand) $);
\coordinate [label=right:\B] (B) at ($ (1.25,0.25) + .1*(rand,rand) $);
\draw [input] (A) -- (B);
\node [name path=D,help lines,draw,label=left:\D] (D) at (A) [circle through=(B)] {};
\node [name path=E,help lines,draw,label=right:\E] (E) at (B) [circle through=(A)] {};
\path [name intersections={of=D and E,by={ [label=above:\C] C}}];
\draw [output] (A) -- (C) -- (B);
\foreach \point in {A,B,C}
\fill [black,opacity=.5] (\point) circle (2pt);
\begin{pgfonlayer}{background}
\fill[triangle!80] (A) -- (C) -- (B) -- cycle;
\end{pgfonlayer}
\node [below right, text width=10cm,align=justify] at (4,3) {
\small\textbf{Proposition I}\par
\emph{To construct an \textcolor{triangle}{equilateral triangle}
on a given \textcolor{input}{finite straight line}.}
\par\vskip1em
Let \A\B be the given \textcolor{input}{finite straight line}. \dots
};
\end{tikzpicture}

```

9 スタイルの設定

- `\tikzset`

大域的に特定のスタイルを定義する. `\tikzset{スタイル名/.style={各種設定}}`とプリアンブルなどに記述するか `tikzpicture` や `scope` 環境の引数として `[スタイル名/.style={各種設定}]` と記述する. また, デフォルト値を設定した上でパラメータを用いてその都度変更することもできる.



```
\begin{tikzpicture}[my grid/.style={green!50,ultra thin}]
  \draw[my grid](0,0)grid(3,2);
\end{tikzpicture}
\begin{tikzpicture}[my grid/.style={#1!50,ultra thin},my grid/.default=blue]
  \draw[my grid](0,0)grid(2,2);
  \draw[my grid=red](3,0)grid(5,2);
\end{tikzpicture}
```

10 不透明度の設定

- `draw opacity=value`

線に対する透明度の指定を行う. `value` には 0 から 1 の数値が入り, 1 は完全は不透明, 0 は完全な透明となる. `.5` はその間である.



```
\begin{tikzpicture}[line width=1ex]
  \draw(0,0)--(3,1);
  \filldraw[fill=yellow!75!black,draw opacity=.5](1,0)rectangle(2,1);
\end{tikzpicture}
```

- `opacity=value`

線および塗りつぶしに対する透明度の指定を行う. `value` には 0 から 1 の数値が入り, 1 は完全は不透明, 0 は完全な透明となる. `.5` はその間であり `semitransparent` と同じである. 下の例で分かるように, `fill` だけではなく `draw` に対しても有効であるから上の `draw opacity` は実際には使わない.



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[opacity=.25](0.5,0)rectangle(1.5,.25);
  \draw[opacity=.5,blue,line width=5pt](0,.25)--(1.25,.25);
\end{tikzpicture}
```

また、透明度はキーワードでも指定できる。

– transparent



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[transparent](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– ultra nearly transparent



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[ultra nearly transparent](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– very nearly transparent



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[very nearly transparent](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– nearly transparent



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[nearly transparent](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– semitransparent



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[semitransparent](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– nearly opaque



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[nearly opaque](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– very nearly opaque



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[very nearly opaque](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

– ultra nearly opaque



```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[ultra nearly opaque](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

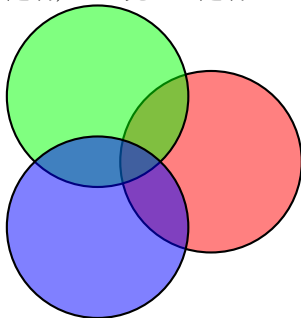
– opaque



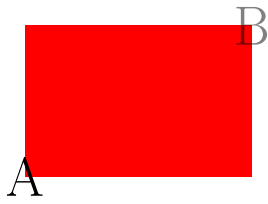
```
\begin{tikzpicture}
  \fill[red](0,0)rectangle(1,0.5);
  \fill[opaque](0.5,0)rectangle(1.5,.25);
\end{tikzpicture}
```

- fill opacity=value

opacity と違い透明度の設定は塗りつぶし部分だけに適用され、線の部分には適用されない。1 は完全は不透明、0 は完全な透明となる。 .5 はその間である。



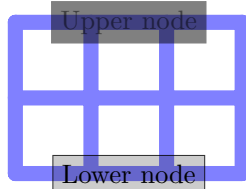
```
\begin{tikzpicture}[thick,fill opacity=.5]
  \filldraw[fill=red](0:1cm)circle[radius=12mm];
  \filldraw[fill=green](120:1cm)circle[radius=12mm];
  \filldraw[fill=blue](-120:1cm)circle[radius=12mm];
\end{tikzpicture}
```



```
\begin{tikzpicture}
\fill[red](0,0)rectangle(3,2);
\node at (0,0){\huge A};
\node[fill opacity=.5]at(3,2){\huge B};
\end{tikzpicture}
```

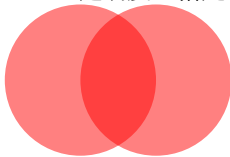
- `text opacity=value`

テキストの透明度を設定する。 `opacity` では図もテキストも全てに透明度を設定するので、場合によってはテキストが見にくくなる。そこで、 `opacity` とは別に `text opacity` でテキストの透明度を設定する。1 は完全是不透明、0 は完全な透明となる。 .5 はその間である。



```
\begin{tikzpicture}[every node/.style={fill,draw}]
\draw[line width=2mm,blue!50,line cap=round](0,0)grid(3,2);
\node[opacity=.5]at(1.5,2){Upper node};
\node[draw opacity=.8,fill opacity=.2,text opacity=1]
at(1.5,0){Lower node};
\end{tikzpicture}
```

ちなみに透明度を指定した図が重なると透明度は加算される。



```
\begin{tikzpicture}[fill opacity=.5]
\fill[red](0,0)circle[radius=1cm];
\fill[red](1,0)circle[radius=1cm];
\end{tikzpicture}
```

これを防ぐ方法が次の “Transparency Groups” である。

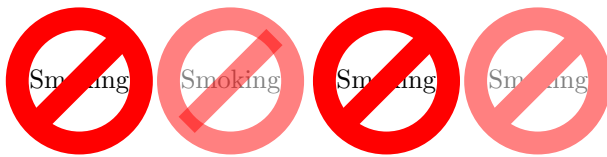
11 Transparency Groups

- transparency group

複数の図をグループ登録し透過度を揃える。



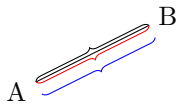
```
\begin{tikzpicture}[opacity=.5]
  \draw[line width=5mm](0,0)--(2,2);
  \draw[line width=5mm](2,0)--(0,2);
\end{tikzpicture}
\begin{tikzpicture}[opacity=.5]
  \begin{scope}[transparency group]
    \draw[line width=5mm](0,0)--(2,2);
    \draw[line width=5mm](2,0)--(0,2);
  \end{scope}
\end{tikzpicture}
```



```
\begin{tikzpicture}
  \node at(0,0)[forbidden sign,line width=2ex,draw=red,fill=white]{Smoking};
  \node [opacity=.5]at(2,0)
    [forbidden sign,line width=2ex,draw=red,fill=white]{Smoking};
\end{tikzpicture}
\begin{tikzpicture}
  \node at(0,0)[forbidden sign,line width=2ex,draw=red,fill=white]{Smoking};
  \begin{scope}[opacity=.5,transparency group]
    \node at(2,0)[forbidden sign,line width=2ex,draw=red,fill=white]{Smoking};
  \end{scope}
\end{tikzpicture}
```

12 その他

- decoration=keyword



```
\begin{tikzpicture}
\node (a){A};
\node (b)at(2,1){B};
\draw(a)--(b);
\draw[decorate,decoration=brace](a)--(b);
\draw[decorate,decoration={brace,mirror},red](a)--(b);
\draw[decorate,decoration={brace,mirror,raise=5pt},blue](a)--(b);
\end{tikzpicture}
```

- `x=value, y=value, z=value`

それぞれ x , y , z 軸方向の単位ベクトルを指定する．長さだけの場合は絶対値だけ変更し，ベクトルで指定するとそのベクトルがそれぞれ x , y , z 軸方向の単位ベクトルとなる． x , y , z の default はそれぞれ 1cm , 1cm , -3.85mm である．

```
\begin{tikzpicture}
\draw(0,0)--+(1,0);
\draw[x=2cm,color=red](0,.5)--+(1,0);
\end{tikzpicture}
```



```
\tikz\draw[x=1.5cm](0,0)grid(2,2);
```

```
\begin{tikzpicture}
\draw(0,0)--(1,1);
\draw[x={(2cm,.5cm)},color=red](0,0)--(1,1);
\end{tikzpicture}
```

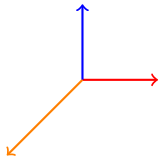


```
\begin{tikzpicture}[smooth]
```

```

\draw plot coordinates{(1,0)(2,.5)(3,0)(3,1)};
\draw[x={(0cm,1cm)},y={(1cm,0cm)},color=red] plot coordinates{(1,0)(2,.5)(3,0)(3,1)};
\end{tikzpicture}

```



```

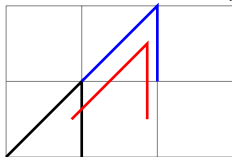
\begin{tikzpicture}[z=-1cm,->,thick]
\draw[color=red](0,0,0)--(1,0.0);
\draw[color=blue](0,0,0)--(0,1,0);
\draw[color=orange](0,0,0)--(0,0,1);
\end{tikzpicture}

```

13 座標の変換

- `shift=coordinate`, `xshift=dimension`, `yshift=dimension`

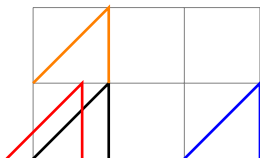
アイテムをそれぞれ指定した分だけ移動させる。



```

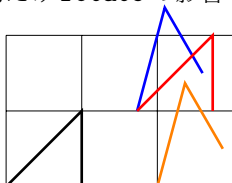
\begin{tikzpicture}[line width=1pt]
\draw[very thin,gray](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[shift={(1,1)},blue](0,0)--(1,1)--(1,0);
\draw[shift={(30:1cm)},red](0,0)--(1,1)--(1,0);
\end{tikzpicture}

```



- `shift only`

基準点への `rotate` は維持するが、その後のアイテムへの `rotate` を無効化する。 `shift` での基準点の移動だけ `rotate` の影響下に置くという感じ。



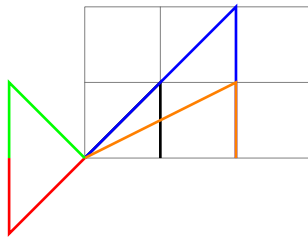
```

\begin{tikzpicture}[line width=1pt]
\draw[very thin](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[rotate=30,xshift=2cm,blue](0,0)--(1,1)--(1,0);
\draw[xshift=2cm,rotate=30,orange](0,0)--(1,1)--(1,0);
\draw[rotate=30,xshift=2cm,shift only,red](0,0)--(1,1)--(1,0);
\end{tikzpicture}

```

- `scale=factor`, `xscale=factor`, `yscale=factor`

基準点を中心としてそれぞれ全方位, x 軸方向, y 軸方向に `factor` 倍する.



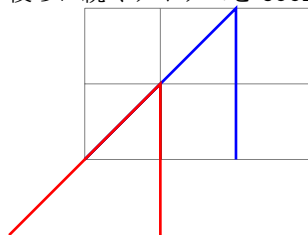
```

\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[scale=2,blue](0,0)--(1,1)--(1,0);
\draw[scale=-1,red](0,0)--(1,1)--(1,0);
\draw[xscale=2,orange](0,0)--(1,1)--(1,0);
\draw[xscale=-1,green](0,0)--(1,1)--(1,0);
\end{tikzpicture}

```

- `scale around={factor:coordinate}`

後ろに続くアイテムを `coordinate` を中心にして `factor` 倍する.



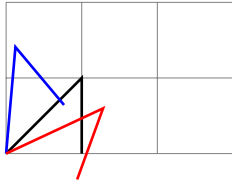
```

\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[scale=2,blue](0,0)--(1,1)--(1,0);
\draw[scale around={2:(1,1)},red](0,0)--(1,1)--(1,0);
\end{tikzpicture}

```

- `rotate=degree`

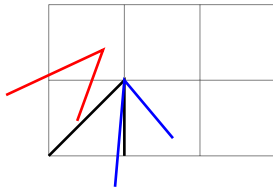
基準点を中心に `degree` 回転させる (60 分法).



```
\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[rotate=40,blue](0,0)--(1,1)--(1,0);
\draw[rotate=-20,red](0,0)--(1,1)--(1,0);
\end{tikzpicture}
```

- `rotate around={degree:coordinate}`

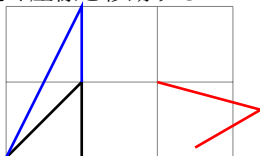
後ろに続くアイテムを `coordinate` を中心として `degree` 回転させる (60 分法).



```
\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[rotate around={40:(1,1)},blue](0,0)--(1,1)--(1,0);
\draw[rotate around={-20:(45:2*sqrt(2))},red](0,0)--(1,1)--(1,0);
\end{tikzpicture}
```

- `cm={a,b,c,d,coordinate}`

`coordinate` が (t_x, t_y) のとき, 座標 (x, y) が $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ によって移される変換で後ろに続く座標を移動する.















```
\begin{tikzpicture}[line width=1pt]
\draw[gray,very thin](0,0)grid(3,2);
\draw(0,0)--(1,1)--(1,0);
\draw[cm={1,1,0,1,(0,0)},blue](0,0)--(1,1)--(1,0);
\draw[cm={cos(60),-sin(60),sin(60),cos(60),(2,1)},red](0,0)--(1,1)--(1,0);
\end{tikzpicture}
```

- `reset cm`

cm で設定された行列を単位行列にする。これは現在のスコープだけでなく大域的に適用されるので、自分が何を行っているのか理解しているのでなければ使ってはいけない。基本的に cm は上の例のような使い方や scope 環境での指定だと思うので、reset cm は使うことはないだろう。

14 塗りつぶし

- 黒・青・赤で以下のパターンが使える。

| | |
|---|---|
|  | <code>\tikz \fill[pattern=horizontal lines](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=vertical lines](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=north east lines](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=north west lines](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=grid](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=crosshatch](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=dots](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=crosshatch dots](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=fivepointed stars](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=sixpointed stars](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=bricks](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |
|  | <code>\tikz \fill[pattern=checkerboard](0,0)--(1,0)--(1,.5)--(0,.5)--cycle;</code> |

15 繰り返し処理

- `\foreach [option] variable in list commands`

list の先頭から最後まで順に variable を代入して commands を実行する。

```
[1][2][3][0] \foreach \x in {1,2,3,0} {[ \x]}
```

list は TeX の制御綴も利用でき、`\expandafter` などで `\foreach` の前に展開する必要はない。

```
[1][2][3][0]
```

```
\def\templist{1,2,3,0}
```

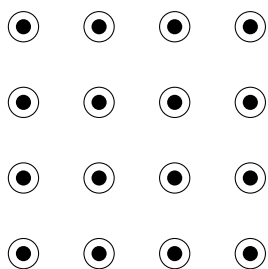
```
\foreach \x in \templist {[ \x]}
```



```
\tikz \foreach \x in {0,1,2,3}
```

```
\draw (\x,0) circle [radius=2mm];
```

`\foreach` を入れ子にすることも可能である。

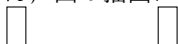


```
\begin{tikzpicture}
\foreach \x in {0,1,2,3}
\foreach \y in {0,1,2,3}
{
\draw (\x,\y) circle [radius=2mm];
\fill (\x,\y) circle [radius=1mm];
}
\end{tikzpicture}
```

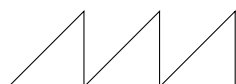
- ... 記法

list には... で省略した書き方も可能である。そしてこの省略記法は非常にインテリジェントである。
 0,1,2,3,4,5,6, \foreach \x in {0,1,...,6} {\x,}
 1,3,5,7,9,11, \foreach \x in {1,3,...,11} {\x,}
 1,3,5,7,9, \foreach \x in {1,3,...,10} {\x,}
 0,0.1,0.20001,0.30002,0.40002, \foreach \x in {0,0.1,...,.5} {\x,} ただし、0.1 は 2 進法では無限小数となるので小数の計算は苦手である。0.5, 0.25, 0.125 など大丈夫かな？

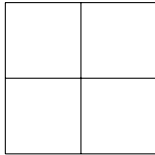
0,0.125,0.25,0.375,0.5, \foreach \x in {0,0.125,...,.5} {\x,}
 9,8,a,b,c,d,e,f,g,2,2.125,2.25,2.375,2.5, \foreach \x in {9,8,a,b,...,g,2,2.125,...,2.5} {\x,}
 $2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7$, \foreach \x in {2¹,2²...,2⁷} { $\$ \x \$$,}
 $0\pi, .5\pi, 1\pi, 1.5\pi, 2\pi, 2.5\pi, 3\pi$, \foreach \x in {0\pi,.5\pi,...\pi,3\pi} { $\$ \x \$$,}
 $A_1, B_1, C_1, D_1, E_1, F_1, G_1, H_1$, \foreach \x in {A_1,..._1,H_1} { $\$ \x \$$,}
 $a_1, a_2, a_3, a_4, a_5, a_6$, \foreach \x in {a_1,a_...,a_6} { $\$ \x \$$,}
 もちろん、図の描画にも利用できる。



```
\tikz \foreach \position in {(0,0),(1,1),(2,0),(3,1)}
\draw \position rectangle +(.25,.5);
```



```
\tikz \draw (0,0)
\foreach \x in {1,...,3} {--(\x,1)--(\x,0)};
```

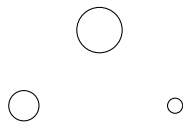


```
\tikz \draw \foreach \p in {1,...,3}
  {(\p,1)--(\p,3)(1,\p)--(3,\p)};
```

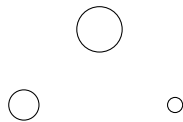
`variable` を `/`(slash) で区切って複数の変数を指定することもできる。変数列中でいくつかの変数の省略が起きた場合は、省略された変数の直前に使用された変数が代用される。

0 1 2 e 3

```
\begin{tikzpicture}
\foreach \x/\xtext in {0,...,3,2.72/e}
\draw (\x,0) node {$\xtext$};
\end{tikzpicture}
```



```
\tikz \foreach \x/\y/\r in {0/0/2mm,1/1/3mm,2/0/1mm}
\draw (\x,\y) circle [radius=\r];
```



```
\tikz \foreach \p/\r in {{(0,0)/2mm},{(1,1)/3mm},{(2,0)/1mm}}
\draw \p circle [radius=\r];
```

• `evaluate` オプション

`\foreach \x in {...}`では`\x`をリストの形式そのままに出力するが、`\foreach \x [evaluate=\x] in {...}`では`\x`の値を計算して出力する。

1.0,2.0,4.0,8.0,16.0,32.0, `\foreach \x [evaluate=\x] in {2^0,2^...,2^5} {\x,}`

また、`\foreach \x [evaluate=\x as \xeval] in {...}`では、`\xeval`に`\x`を計算した結果が代入される。

$2^0 = 1.0, 2^1 = 2.0, 2^2 = 4.0, 2^3 = 8.0, 2^4 = 16.0, 2^5 = 32.0,$

```
\foreach \x [evaluate=\x as \xeval] in {2^0,2^...,2^5} {$\x=\xeval$,}
```

さらに、`using`を利用して計算させることもできる。



```
\tikz \foreach \x [evaluate=\x as \shade using \x*10] in {0,1,...,10}
  \node [fill=red!\shade!yellow,minimum size=0.65cm] at (\x,0) {\x};
```

- remember オプション

`remember=\x as \lastx (initially A)` とすることで、直前で使用された `\x` の値を `\lastx` に代入して利用する。この例の `\lastx` の初期値は A である。

$\overrightarrow{AB}, \overrightarrow{BC}, \overrightarrow{CD}, \overrightarrow{DE}, \overrightarrow{EF}, \overrightarrow{FG}, \overrightarrow{GH},$

```
\foreach \x [remember=\x as \lastx (initially A)] in {B,...,H}
  {\$\overrightarrow{\lastx\x}$,}
```

- count オプション

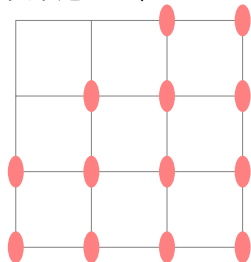
`list` の何番目の元を使用しているかカウントする制御綴を指定する。

| | | | | |
|----|----|----|----|----|
| aa | bb | cc | dd | ee |
| ab | cc | dd | ee | |
| ac | dd | ee | | |
| ad | ee | | | |
| ae | | | | |

```
\tikz [x=.75cm,y=.75cm]
\foreach \x [count=\xcounter] in {a,...,e}
  \foreach \y [count=\ycounter] in {\x,...,e}
    \node [draw,top color=white,bottom color=blue!50,minimum size=.666cm]
      at (\xcounter,-\ycounter) {\$\mathstrut\x\y$};
```

- \breakforeach

文字通りに `\foreach` を中断させる。



```
\begin{tikzpicture}
\foreach \x in {1,...,4}
\foreach \y in {1,...,4}
{
  \fill [red!50] (\x,\y) ellipse [x radius=3pt,y radius=6pt];
  \ifnum \x<\y
    \breakforeach
  \fi
}
```



```

    }
\end{tikzpicture}

```

16 各種計算

16.1 通常の計算

- `\pgfmathparse{add(1,2)}`, `\pgfmathparse{1+2}`
`\pgfmathresult--> 3.0`
- `\pgfmathparse{subtract(10,2)}`, `\pgfmathparse{10-2}`
`\pgfmathresult--> 8.0`
- `\pgfmathparse{neg(4)}`
`\pgfmathresult -4.0`
- `\pgfmathparse{multiply(4,3)}`, `\pgfmathparse{4*3}`
`\pgfmathresult--> 12.0`
- `\pgfmathparse{divide(75,6)}`, `\pgfmathparse{75/6}`
`\pgfmathresult--> 12.5`
- `\pgfmathparse{div(75,6)}`
`\pgfmathresult--> 12`
- `\pgfmathparse{factorial(4)}`, `\pgfmathparse{4!}`
`\pgfmathresult--> 24.0`
- `\pgfmathparse{sqrt(10)}`
`\pgfmathresult--> 3.16227`
- `\pgfmathparse{pow(2,7)}`
`\pgfmathresult--> 128.0`
- `\pgfmathparse{e}`
`\pgfmathresult--> 2.718281828`
- `\pgfmathparse{exp(1)}`
`\pgfmathresult--> 2.71825`
- `\pgfmathparse{ln(e)}`
`\pgfmathresult--> 0.99995`
- `\pgfmathparse{log10(100)}`
`\pgfmathresult--> 1.99997`
- `\pgfmathparse{log2(128)}`
`\pgfmathresult--> 6.99994`
- `\pgfmathparse{abs(-6)}`
`\pgfmathresult--> 6.0`
- `\pgfmathparse{mod(20,6)}`
`\pgfmathresult--> 2.0`
- `\pgfmathparse{Mod(-20,6)}`
`\pgfmathresult--> 4.0`

【注】 割り算の結果の整数部分

16.2 丸め計算

- 小数点以下を切り捨てた整数値を返す.

```
\pgfmathparse{round(6.125)} \pgfmathresult--> 6.0  
\pgfmathparse{round(-6.125)} \pgfmathresult--> -6.0  
\pgfmathparse{round(6)} \pgfmathresult--> 6.0  
\pgfmathparse{round(-6)} \pgfmathresult--> -6.0
```

- 引数以下の整数値を返す.

```
\pgfmathparse{floor(6.625)} \pgfmathresult--> 6.0  
\pgfmathparse{floor(-6.625)} \pgfmathresult--> -7.0  
\pgfmathparse{floor(6)} \pgfmathresult--> 6.0  
\pgfmathparse{floor(-6)} \pgfmathresult--> -7.0
```

- 引数以上の整数値を返す.

```
\pgfmathparse{ceil(6.625)} \pgfmathresult--> 7.0  
\pgfmathparse{ceil(-6.625)} \pgfmathresult--> -6.0  
\pgfmathparse{ceil(6)} \pgfmathresult--> 6.0  
\pgfmathparse{ceil(-6)} \pgfmathresult--> -6.0
```

- 引数の整数部分を返す.

```
\pgfmathparse{int(6.625)} \pgfmathresult--> 6  
\pgfmathparse{int(-6.625)} \pgfmathresult--> -6  
\pgfmathparse{int(6)} \pgfmathresult--> 6  
\pgfmathparse{int(-6)} \pgfmathresult--> -6
```

- 引数の小数部分を返す.

```
\pgfmathparse{frac(6.625)} \pgfmathresult--> 0.625  
\pgfmathparse{frac(-6.625)} \pgfmathresult--> 0.625  
\pgfmathparse{frac(6)} \pgfmathresult--> 0.0  
\pgfmathparse{frac(-6)} \pgfmathresult--> 0.0
```

16.3 三角関数関連

- `\pgfmathparse{pi}` `\pgfmathresult--> 3.141592654`
- `\pgfmathparse{pi r}` `\pgfmathresult--> 179.99962`

【注】数値の直後に `r` をつけると弧度法から 60 分法に変換する.

- `\pgfmathparse{rad(180)}` `\pgfmathresult--> 3.14159`
- `\pgfmathparse{deg(pi)}` `\pgfmathresult--> 179.99962`
- `\pgfmathparse{sin(60)}` `\pgfmathresult--> 0.86603`
`\pgfmathparse{sin(pi/3 r)}` `\pgfmathresult--> 0.86601`
- `\pgfmathparse{cos(60)}` `\pgfmathresult--> 0.5`
`\pgfmathparse{cos(pi/3 r)}` `\pgfmathresult--> 0.49998`
- `\pgfmathparse{tan(60)}` `\pgfmathresult--> 1.73206`
`\pgfmathparse{tan(pi/3 r)}` `\pgfmathresult--> 1.73203`
- `\pgfmathparse{sec(60)}` `\pgfmathresult--> 2.0`
`\pgfmathparse{sec(pi/3 r)}` `\pgfmathresult--> 2.0`

- `\pgfmathparse{cosec(60)}` `\pgfmathresult--> 1.15463`
`\pgfmathparse{cosec(pi/3 r)}` `\pgfmathresult--> 1.15472`
- `\pgfmathparse{cot(60)}` `\pgfmathresult--> 0.57732`
`\pgfmathparse{cot(pi/3 r)}` `\pgfmathresult--> 0.57733`
- `\pgfmathparse{asin(.5)}` `\pgfmathresult--> 30.0`
`\pgfmathparse{rad(asin(.5))}` `\pgfmathresult--> 0.52359`
- `\pgfmathparse{acos(.5)}` `\pgfmathresult--> 60.0`
`\pgfmathparse{rad(acos(.5))}` `\pgfmathresult--> 1.0472`
- `\pgfmathparse{atan(1)}` `\pgfmathresult--> 45.0`
`\pgfmathparse{rad(atan(1))}` `\pgfmathresult--> 0.78539`
- `\pgfmathparse{atan2(1,sqrt(3))}` `\pgfmathresult--> 60.01506`
`\pgfmathparse{rad(atan2(1,sqrt(3)))}` `\pgfmathresult--> 1.04745`

【注】 `atan2(x,y)` は原点と (x, y) を結んだ直線と x 軸のなす角度。

16.4 論理関数

- `\pgfmathequal{x}{y}` `\pgfmathparse{equal(x,y)}`
 $x = y$ ならば 1, そうでなければ 0 を返す.
- `\pgfmathgreater{x}{y}` `\pgfmathparse{greater(x,y)}`
 $x > y$ ならば 1, そうでなければ 0 を返す.
- `\pgfmathless{x}{y}` `\pgfmathparse{less(x,y)}`
 $x < y$ ならば 1, そうでなければ 0 を返す.
- `\pgfmathnotequal{x}{y}` `\pgfmathparse{notequal(x,y)}`
 $x \neq y$ ならば 1, そうでなければ 0 を返す.
- `\pgfmathnotgreater{x}{y}` `\pgfmathparse{notgreater(x,y)}`
 $x \leq y$ ならば 1, そうでなければ 0 を返す.
- `\pgfmathnotless{x}{y}` `\pgfmathparse{notless(x,y)}`
 $x \geq y$ ならば 1, そうでなければ 0 を返す.
- `\pgfmathand{A}{B}` `\pgfmathparse{and(A,B)}`
条件 A, B がともに真ならば 1, そうでなければ 0 を返す.
- `\pgfmathor{A}{B}` `\pgfmathparse{or(A,B)}`
条件 A, B のどちらかが真ならば 1, そうでなければ 0 を返す.
- `\pgfmathnot{A}` `\pgfmathparse{not(A)}`
条件 A が偽ならば 1, そうでなければ 0 を返す.
- `\pgfmathifthenelse{A,X,Y}` `\pgfmathparse{ifthenelse(A,X,Y)}`
条件 A が真ならば X , そうでなければ Y を返す.
- `\pgfmathtrue` `\pgfmathparse{true}`
常に 1 を返す.
- `\pgfmathfalse` `\pgfmathparse{false}`
常に 0 を返す.

16.5 ランダム関数

- `\pgfmathrnd` `\pgfmathparse{rnd}`

0 以上 1 以下の数値を返す.

- `\pgfmathrand \pgfmathparse{rand}`

-1 以上 1 以下の数値を返す.

- `\pgfmathrandom{m,n} \pgfmathparse{random(m,n)}`

この関数は引数を 0 個または 1 個または 2 個とり, 0 個の場合は`\pgfmathrnd` と同じく 0 以上 1 以下の数値を返す. `\pgfmathrandom{m}` のように 1 個の場合は 1 以上 m 以下の整数値を返す. `\pgfmathrandom{m,n}` のように 2 個の場合は m 以上 n 以下の整数値を返す.

16.6 底の変換関数

- `\pgfmathhex{x} \pgfmathparse{hex(x)}`

10 進法から 16 進法へ変換する (小文字).

- `\pgfmathHex{x} \pgfmathparse{Hex(x)}`

10 進法から 16 進法へ変換する (大文字).

- `\pgfmathoct{x} \pgfmathparse{oct(x)}`

10 進法から 8 進法へ変換する.

- `\pgfmathbin{x} \pgfmathparse{bin(x)}`

10 進法から 2 進法へ変換する.

16.7 その他の関数

- `\pgfmathparse{min(a,b,c,...)}`

引数内で最小の数値を返す.

`-2.0 \pgfmathparse{min(-2,-1,0,1,2,)}\pgfmathresult$`

- `\pgfmathparse{max(a,b,c,...)}`

引数内で最大の数値を返す.

`2.0 \pgfmathparse{max(-2,-1,0,1,2,)}\pgfmathresult$`

- `\pgfmathveclen{x}{y} \pgfmathparse{veclen(x,y)}`

ベクトル (x, y) の長さを返す.

`12.99976 \pgfmathparse{veclen(5,12)}\pgfmathresult`

- `\pgfmatharray{x}{n} \pgfmathparse{array(x,n)}`

配列 x の n 番目の要素を返す (配列は 0 番目から始まる).

`1 \pgfmathparse{array({-2,-1,0,1,2},3)}\pgfmathresult`

- `\pgfmathsinh{x} \pgfmathparse{sinh(x)}`

双曲線正弦関数の値を返す. $\sinh x = \frac{e^x - e^{-x}}{2}$.

- `\pgfmathcosh{x} \pgfmathparse{cosh(x)}`

双曲線余弦関数の値を返す. $\cosh x = \frac{e^x + e^{-x}}{2}$.

- `\pgfmathtanh{x} \pgfmathparse{tanh(x)}`

双曲線正弦関数の値を返す. $\tanh x = \frac{\sinh x}{\cosh x}$.

- `\pgfmathwidth{"strings"} \pgfmathparse{width("strings")}`

`strings` の長さを返す.

- `\pgfmathheight{"strings"} \pgfmathparse{height("strings")}`

`strings` の高さを返す.

- `\pgfmathdepth{"strings"} \pgfmathparse{depth("strings")}`
strings の深さを返す.

16.8 新しい関数の定義

与えられた引数の 2 倍の値を返す関数 `double` は次のように定義して使用する. この例では `\pgfmathresult` の値は 88.6 である.

```
\makeatletter
\pgfmathdeclarefunction{double}{1}{
  \begingroup
    \pgf@x=#1pt\relax
    \multiply\pgf@x by2\relax
    \pgfmathreturn\pgf@x
  \endgroup
}
\makeatother
\pgfmathparse{double(44.3)}\pgfmathresult
```

この定義で同時に `\pgfmathdouble{x}` も定義される. また, `\pgfmathdeclarefunction` における定義では `\pgf@x` の他に `\pgf@y`, `\c@pgf@counta`, `\c@pgf@countb` がある.

