

The Mindmap Module

Li Yanrui (lyr.m2@live.cn)

<https://github.com/liyanrui/snail>

Introduction	1.	3	New Root	4.
1 Branches	2.	4	Branch Quote	4.
2 Style	3.		Afterwords	6.

Introduction

The mindmap is a ConTeXt module written in MetaPost for drawing mind maps. Unlike most mind-mapping software, the mindmap module places all information on paths, its nodes carry no content at all. In other words, the module sees no container-like nodes filled with text or images. A mind map is simply a set of connected paths, and the information appears as annotations along them.

The simplest ConTeXt source file for using the mindmap module is as follows.

```
\usemodule[mindmap]
\startMPpage

% some MetaPost code for drawing mind map.

\stopMPpage
```

Create a source file named `foo.tex` that its content is

```
\usemodule[mindmap]
\startMPpage
mind.enter("$\delta_{ij}$ is", 15);
  mind("1, if  $i=j$ .", 15);
  mind("0, otherwise.", -15);
mind.exit;
\stopMPpage
```

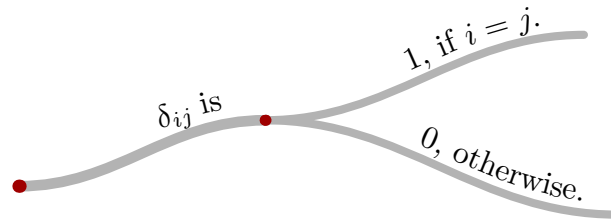
Use the `context` command to compile it into `foo.pdf` in the same directory.

```
$ context foo.tex

or

$ context foo
```

Then you can get the following result as shown in Example 1.



Example 1 First mind map

1 Branches

Every thought of yours can be expressed as a single branch in a mind map—just keep it as concise as possible, for instance:

```
mind("$\delta_{ij}$", 15);
```

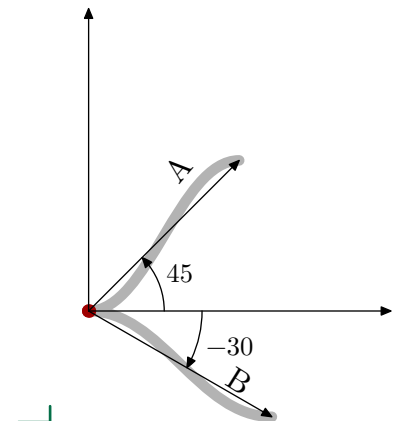
In the code above, the 15 is the angle that indicates the direction of the branch in map. Every branch need an angle degree like this. The example below can help you understanding these.

```
pair base, a, b;
base := mindmap.currentbase;
mind("A", 45); a := mindmap.currentend;
mind("B", -30); b := mindmap.currentend;

path pa, pb, ox, oy;
pa := base -- a; pb := base -- b;
ox := base -- (base + (4cm, 0));
oy := base -- (base + (0, 4cm));

path angle_a, angle_b;
anglelength := 1cm;
angle_a := anglebetween(ox, pa, "\tex 45$");
anglelength := 1.5cm;
angle_b := anglebetween(ox, pb, "\tex -30$");

for it = pa, pb, ox, oy, angle_a, angle_b:
    drawarrow it;
endfor;
```



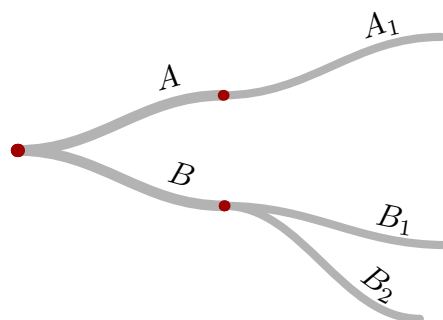
Example 2 Angles in mind map

If a branch has some deeper ones, you need to enter it and create child branches for it. When you want to go back to the parent branch and start new thought in the same level, you must exit from current child branches; see the following example.

```

mind.enter("$A$", 15);
  mind("$A_{1}$", 15);
mind.exit;
mind.enter("$B$", -15);
  mind("$B_{1}$", -15);
  mind("$B_{2}$", -30);
mind.exit;

```



Example 3 Entering and exiting branch

2 Style

The thickness of each branch decreases as the branch level increases. The top-level branch thickness defaults to **4pt**, but this can be changed with the **mind.thickness** macro. For the n -th level, the branch thickness equals the top-level thickness divided by 1.3^n .

By default, all branches are colored darkgray, but the macro **mind.colors** can be used to assign a specific color to each level's branches. The colors of branch knots can be controlled with the **mind.knotcolor** macro.

The example below sets the thickness of first level branch to **6pt**, and assigns colors to the branches and knots of levels 1 to 3.

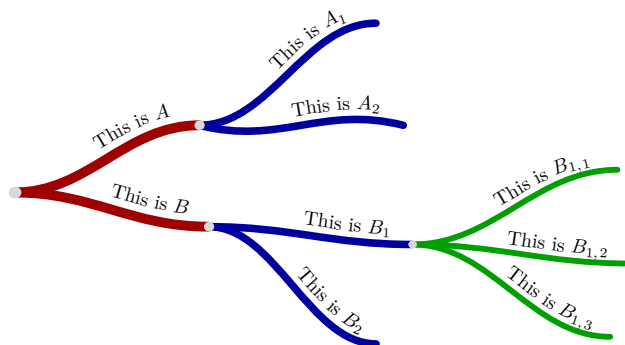
```

mind.thickness(6pt);
mind.colors(darkred,
            darkblue,
            darkgreen);
mind.knotcolor(lightgray);

mind.enter("This is $A$", 20);
  mind("This is $A_1$", 30);
  mind("This is $A_2$", 0);
mind.exit;

mind.enter("This is $B$", -10);
  mind.enter("This is $B_{1}$", -5);
    mind("This is $B_{1,1}$", 20);
    mind("This is $B_{1,2}$", -5);
    mind("This is $B_{1,3}$", -25);
  mind.exit;
  mind("This is $B_2$", -35);
mind.exit;

```



Example 4 Branch style setting

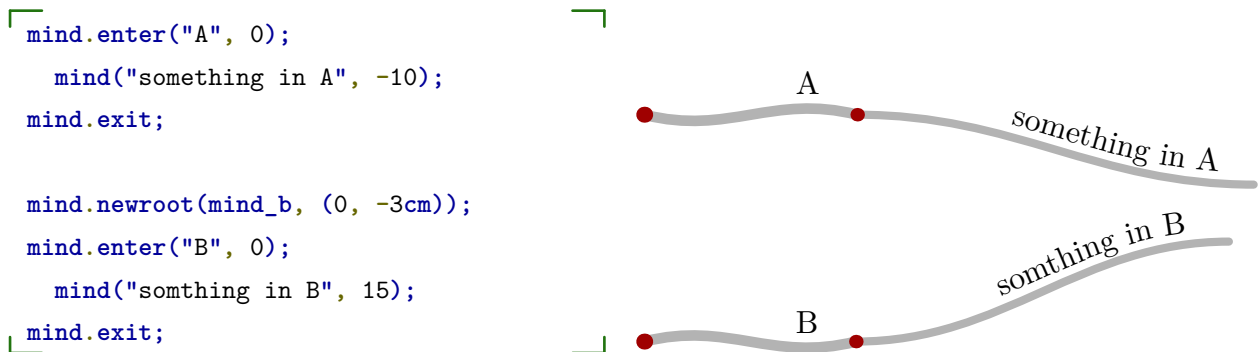
If you feel the lengths of branches too short, you can stretch them by a given factor using the `mind.stretch` macro. The example below stretches the branches to twice their default length.



Example 5 Stretching branches

3 New Root

Mind maps drawn by the mindmap module are not strictly tree-structured. The default root is at `(0, 0)`, but you can use the `mind.newroot` macro to create the starting point or root of new mind map. For instance,



Example 6 New root

The `mind_b` in the code above is a variable of MetaPost's `pair` type, that stores the location of the new root.

4 Branch Quote

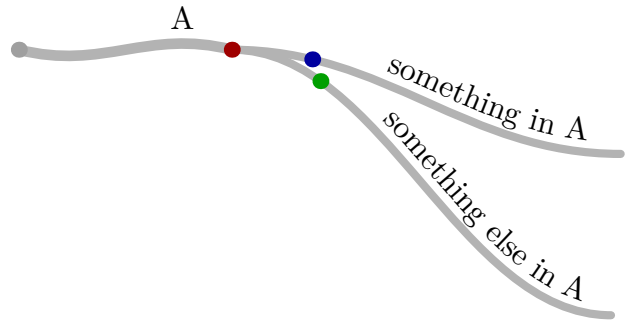
Once you create a new branch, you can catch its base and handle point with the macros `mind.base` and `mind.handle`.

```

pair anchor[];
mind.knotcolor(darkgray);
mind.enter("A", 0);
  mind("something in A", -15);
  mind.handle(anchor1);
  mind("something else in A", -35);
  mind.handle(anchor2);
  mind.base(anchor3);
mind.exit;

pickup pencircle scaled 6pt;
draw anchor1 withcolor darkblue;
draw anchor2 withcolor darkgreen;
draw anchor3 withcolor darkred;

```



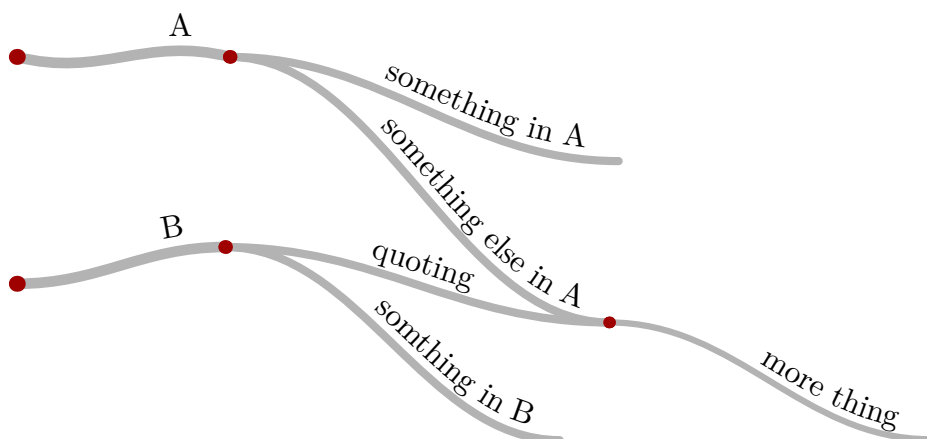
Example 7 Anchors

Based on these anchor points, we can quote a branch in other tree with a new root. The following example shows a scenario where two trees share a branch.

```

pair demo;
mind.enter("A", 0);
  mind("something in A", -15);
  mind.enter("something else in A", -35);
    mind("more thing", -20); mind.base(demo);
  mind.exit;
mind.exit;
mind.newroot(B, (0, -3cm));
mind.enter("B", 10);
  mind.quote("quoting", demo);
  mind("something in B", -30);
mind.exit

```



Example 8 Quoting branch

Afterwords

The mindmap module is a practice in learning the MetaPost language. Its inspiration and foundation come from the macro `lmt_followtext`, implemented by Hans Hagen in LuaMetaFun—the next-generation MetaPost still under development—which places text along an arbitrary curved path; see Chapter 5 of the LuaMetaFun manual. Within the ConTeXt LMTX environment, the manual can be founded with the following command:

```
$ mtxrun --search luametafun.pdf
```