Flow – a program to generate flowcharts in the LaTeX picture environment

Terry Brown

November 25, 1992 — September 22, 2005

Summary

Flow is a small program which parses the flowchart description explained in this document and translates it to the LaTeX picture environment.

Flow acts as a filter, so a typical execution would be...

```
flow <myprog.flo >myprog.pic
```

...where `myprog.flo` is a plain text file containing a description of a flowchart, and `myprog.pic` will be a plain text file containing the body of a LaTeX picture environment to draw the flowchart. Alternatively

`flow inputFile` or `flow inputFile outputFile`

may be used. VMS users see notes at end.

`myprog.pic` could either be inserted into a picture environment in a LaTeX file with a text editor, or pulled in by the `\input` command as LaTeX is run.

System requirements

Nothing in particular. Flow is in very plain C and should be portable to pretty much anything without alteration. Check beginning of file for defines if it gives trouble.

Distribution and Bug Reports

Flow is free software covered by the GPL. See the file `COPYING` for details. Email to `terry_n_brown@yahoo.com` with bug reports etc.

Flowchart description syntax

`[indented text]` indicates the command accepts zero or more line of text, these lines are identified by indentation. All lines after the command starting with either a space or a tab are assumed to be text for that command. The first line not starting with one of these characters is assumed to be the next command. Distributed code / executeables are case insensitive, but local compilations may vary.

The flowchart is always "going" either up, down, left or right. The initial direction is down.

The flowchart description file is terminated either by an invalid command, a blank line, or the end of the file. Keywords are case sensitive.

The output from flow will appear inside a LaTeX picture environment. The positioning of the picture can be adjusted with the second pair as usual for the picture environment.

`\unitlength` should be set for use with the picture environment, all the examples in this document use `2em` (ie. put `\setlength{\unitlength}{2em}` at the top of the LaTeX file). Smaller values make the boxes tighter around the text, larger values make them more open.

The commands

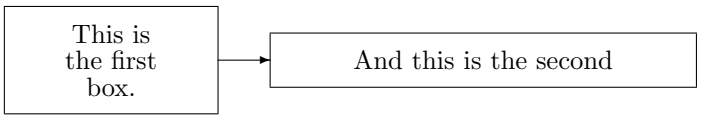
```
% comment to end of line
```

If the first thing on a line is a '`%`' symbol the line is treated as a comment. Thanks to Joost Bruynooghe for adding this command.

```
Box [x y]
  [indented text]
```

Draws a text at the current position, including a line or arrow leading to it if appropriate. If x and y are specified, the size of the box (in `\unitlengths`), and all subsequent boxes, is set to these. The default size of a box is 4 by 2 `\unitlengths`.

```
Right
Box
  This is
  the first
  box.
Box 8 1
  And this is the second
```



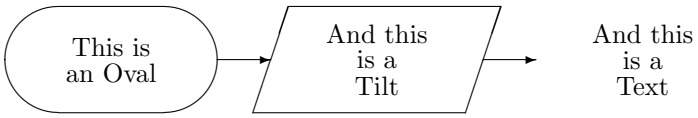
```
Oval [x y]
  [indented text]

Tilt [x y]
  [indented text]

Text [x y]
  [indented text]
```

`Oval`, `Tilt` and `Text` are identical to `Box`, except for the shape of the frame. (`Text` is an invisible frame.)

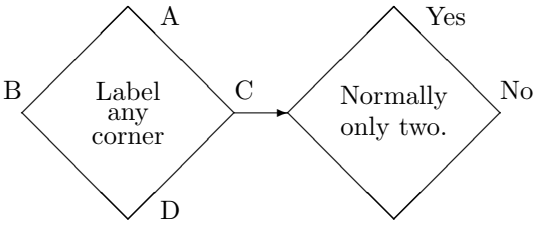
```
Right
Oval
  This is
  an Oval
Tilt
  And this
  is a
  Tilt
Text
  And this
  is a
  Text
```




```
Choice A B C D [x y]
  [indented text]
```

Draws a choice diamond, with the corners labeled as illustrated by the left diamond in the example. Periods (.) are not printed. The optional x an y parameters alter the size of the choice. The default size is 4 by 4 `\unitlengh`s. Flow will report an error for non-square choice boxes whose aspect ratio doesn't match one of the line slopes supported by LaTeX (-6 – 6 : -6 – 6, integers only).

```
Right
Choice A B C D
  Label
  any
  corner
Choice Yes . No .
  Normally
  only two.
```



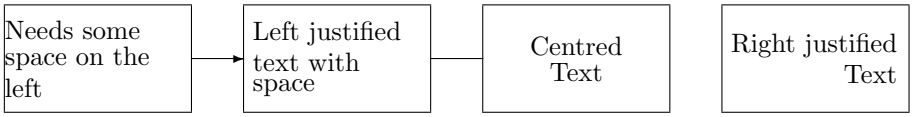
`SetTrack none | arrow | line`

Use arrows, lines, or nothing for drawing connections between boxes.

`TxtPos P1 P2 [B [A]]`

`P1` is the LaTeX position specification (eg. [c] or [l]) for the lines of text that makes up the blocks of text in the boxes, `P2` is the LaTeX position specification for the whole block of text within the box. `B` is the string (no white space) to be placed before each line of text, `A` is the string to be placed after each line of text. The example shows the use of `B` to keep text off the edge of the box.

```
Right
SetTrack arrow
TxtPos [l] [l]
Box 3.5 2
  Needs some
  space on the
  left
TxtPos [l] [l] ~
Box
  Left justified
  text with
  space
SetTrack line
TxtPos [c] [c]
Box
  Centred
  Text
SetTrack none
TxtPos [r] [r] ~ \hspace*{1ex}
Box
  Right justified
  Text
```

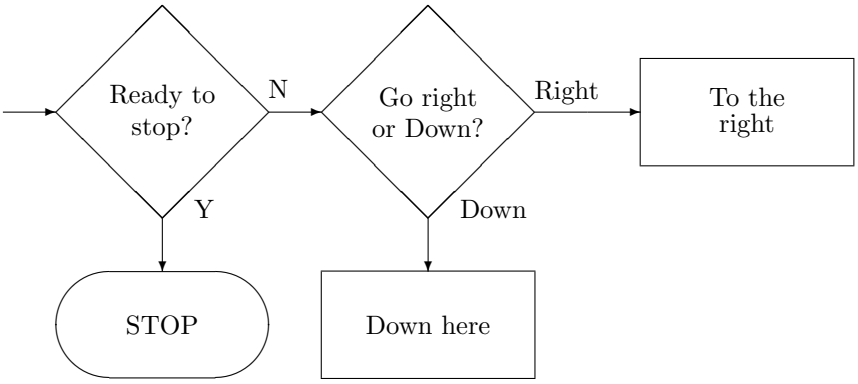


`Tag`

`ToTag`

`Tag` stores the location and size of the last object drawn on a stack, `ToTag` returns to that position (removing the item from the stack). This is particularly useful with `Choice`s, allowing a second chain to be built from the diamond, but it can be used with any other item. Flow will complain if it encounters more `ToTag`s the `Tag`s, but won't mention `Tag`s left on the stack when it finishes.

```
Right 0
Choice . . N Y
  Ready to
  stop?
Tag
Choice . . Right Down
  Go right
  or Down?
Tag
Right 1
Box
  To the
  right
ToTag
Down
Box
  Down here
ToTag
Down
Oval
  STOP
```

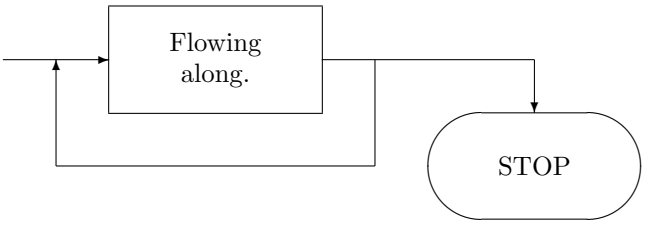


```
Up    [d [*]]
Down  [d [*]]
Left  [d [*]]
Right [d [*]]
```

With or without the optional parameter, these command change the current direction of the flowchart. With the optional parameter, they draw a line, if `SetTrack` is `line` *or* `arrow`, or leave a gap, if `SetTrack` in `none`. The length of the line (or gap) is *d*. By default `Box`es, `Oval`s, `Tilt`s and `Text`s are 4 units wide and 2 units high. A `Choice` is 4 by 4 units. The arrows that connect things together are 1 unit. So a box drawn while the current direction is down would occupy $2 + 1 = 3$ vertical units.

To force the line to end in a arrow head, use the "*", which must be separated from the "d" by a space.

```
Right 1
Box
  Flowing
  along.
Right 1
Tag
Down 2
Left 6
Up 2 *
ToTag
Right 3
Down
Oval
  STOP
```

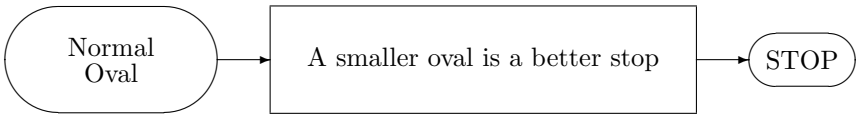


`Scale x y`

Scale the *next* item by the specified values

```
Right
Oval
  Normal
  Oval
Scale 2 1
Box
  A smaller oval is a better stop
Scale 0.5 0.5
```

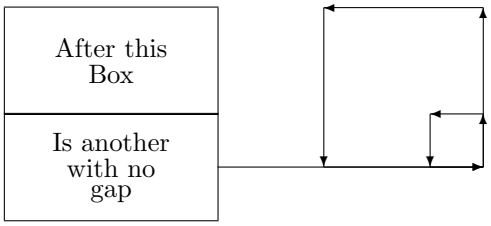
```
Oval
  STOP
```




```
Skip x0 y0 x1 x1
```

- `x0` Horizontal separation between boxes, default 1 `\unitlength`
- `y0` Vertical separation between boxes, default 1 `\unitlength`
- `x1` Multiplier for `Left` and `Right` commands, default 1
- `y1` Multiplier for `Up` and `Down` commands, default 1

```
Skip 0 0 1 1
SetTrack none
Box
  After this
  Box
Box
  Is another
  with no
  gap
SetTrack arrow
Right 5
Up 1 *
Left 1 *
Down 1 *
Right 1 *
Skip 0 0 3 3
Up 1 *
Left 1 *
Down 1 *
Right 1 *
```



VMS notes

I don't know much VMS, but this is one way of getting it to work. Compile and link as normal, then

```
flow :== $ $1$DIA3: [brownt1.usr.flow]flow.exe
```

where the bit in the box is the name of the drive you're working on, and `[brownt1.usr.flow]` is the appropriate path. Then use the

```
flow infile outfile
```

form, as the redirection form doesn't seem to work.

An example

The instructions that generated this flow chart are included in a commented section in `flowdoc.tex`. Note the block of text is part of the picture environment (a `Text`).