My Way

January 30, 2007

\sometxt: some nice techniques for placing text labels on graphics

Mojca Miklavec

This will be the abstract - but this is still a DRAFT version of the document. Neither content nor the design has been finished/polished out. Please report any bugs/typos/comments/suggestions.

1 Hello world!

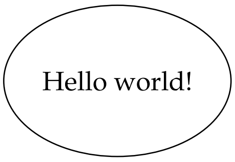
In METAPOST, the standard way of placing text on graphics is to use the btex ... construct as in the following example:

```
beginfig(1);
   draw fullcircle xyscaled (3cm,2cm);
   label(btex Hello world! etex, origin);
endfig;
end.
```

In ConTEXt you would now do the same with::

```
\startMPcode
   draw fullcircle xyscaled (3cm,2cm);
   label(\sometxt{Hello world!}, origin);
\stopMPcode
```

Both approaches should yield equivalent result in this case:



However, the btex ... syntax has so many drawbacks (in both efficiency and flexibility) that it's use is **strongly discouraged**.

There exists yet another possibility to place labels, namely `textext("...")`[1] with the same hello-world example below.

```
\startMPcode
   draw fullcircle xyscaled (3cm,2cm);
   label(textext("Hello world!"), origin);
\stopMPcode
```

Before Spring 2006 it was so buggy that it was hardly working anywhere and very soon after Hans & Taco fixed most bugs, a much more efficient \sometxt has been introduced to replace it. But despite its inefficiency, textext cannot be replaced trivially by \sometxt in all the cases. For auto-generated strings, loops and macros it's still useful.

For an average user there's really not much left to say about \sometxt, apart from asking them to use it instead of any other alternative if possible. The rest of the document is devoted to the curious ones.

[1] documented in the MetaFun manual

2 Some Pitfalls

(I don't like the section name)

Normally this should be the very last section in the manual, but **some default settings need to be changed** if you experience problems (or even before you do).

2.1 \runMPgraphicstrue, \runMPTEXgraphicstrue

To be able to use `\sometxt` properly, you need to have `runMPgraphics` set to `true`.[2]

In order to be able to use `textext` at all you also need to set `\runMPTEXgraphicstrue`.

You'll find the two settings in `tex/context/user/cont-sys.rme` in your *TEXMF* tree. If you want to change these values, uncomment the corresponding two lines and save the file as `cont-sys.tex`. That way your settings won't get lost next time when you update ConTEXt.

TODO: I have to check if one really has to enable write18 in order to be able to use `\runMPTEXgraphicstrue`*. Because that would mean that one can only use* `\sometxt` *with* `write18` *enabled. That would be a pitty.*

[2] This might be a bug, but without it you'll get wrong scaling when using more than one text label in a picture.

2.2 Color stacks

If you want to use `withcolor`, you need[3]

```
\chardef\TeXtextcolormode\zerocount
```

The value of `\TeXtextcolormode` means:

- **0**: nothing, withcolor works ok, but nested colors fail
- **1**: local color stack ok
- **2**: obey color stack (not yet supported)

To see the difference, take a look at the following examples:

```
\startuseMPgraphic{text color mode}
 draw \sometxt{this is \color[red]{red} and blue} withcolor blue;
 draw \sometxt{this is \color[red]{red} and black} shifted (4cm,0);
\stopuseMPgraphic

% default
\chardef\TeXtextcolormode\plusone
\useMPgraphic{text color mode}
\color[green]{\useMPgraphic{text color mode}}

\chardef\TeXtextcolormode\zerocount
\useMPgraphic{text color mode}
\color[green]{\useMPgraphic{text color mode}}
```

this is red and blue	this is red and black
this is red and blue	this is red and black
this is red and blue	this is red and black
this is red and blue	this is red and black

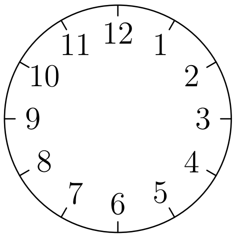
[3] This document has been typeset with `\chardef\TeXtextcolormode\zerocount` as well.

3 textext("not to be thrown away yet")

Most commercials or even manuals tell you only what the product they're selling is good for. Here we'll make an exception and start with something where `\sometxt` is not good at all, so we'll be forced to use the old `textext` macro for it.

On the first figure there is circle drawn first, followed by a loop which places twelve tics and numbers[4] next to them, each one rotated for $i{\cdot}30°$ in the clockwise (negative) direction.

```
\startMPcode
 numeric r; r = 1.5cm;
 draw fullcircle scaled 2r;
 for i=1 upto 12:
  draw (origin--down) scaled 4pt shifted (0,r) rotated -30i;
  label(textext(decimal i), up scaled .75r rotated -30i);
 endfor;
\stopMPcode
```



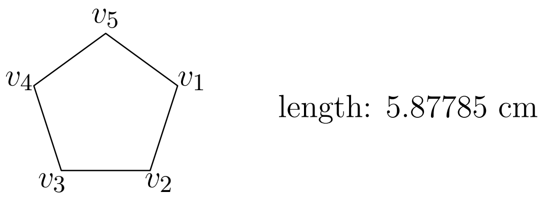
`decimal` converts numerical value of the counter *i* into string, which is then passed to `textext()` as an argument.

Passing values to both `btex ... etex` and `\sometxt` in such a way is not possible, that's why more tricky approaches are needed to achieve the same result. Or, we can always call `btex 1 etex`, `textext("1")` or `\sometxt{1}` twelve times of course, with different argument each time.

[4] Placement of numbers is visually bad, but placing them properly would require longer code and wouldn't bring anything to understanding of the example.

The second example shows us how strings can be concatenated together, so that the counter now takes the role of an index inside a math expression. Also, the calculated length of the perimeter can be written. That last thing would be the least trivial to get with `\sometxt`. It's probably not impossible to do it[5], but this article won't cover it.

```
\startMPcode
 numeric r; r = 1cm;
 pair v[];
 path p;

 for i=1 upto 5:
  % define the five vertices
  v[i] = up scaled r rotated -72i;
  % draw labels: $v_1$, $v_2$, ... $v_5$
  label(textext("$v_" & decimal i & "$"), v[i] scaled 1.2);
 endfor;

 % define and draw the pentagon
 p = for i=1 upto 5: v[i]-- endfor cycle;
 draw p;
 % write out the calculated perimeter
 draw textext("length: " & decimal (arclength p/1cm) & " cm") shifted
(4cm,0);
\stopMPcode
```



[5] If you like challenges, this might be a nice one. If you find a solution, please share it with the ConTEXt community.

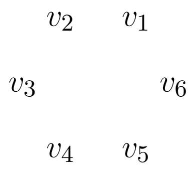
If you want to use `textext` inside your own macros (which possibly reside in an external file), you should also use `\forceMPTEXcheck{your_macro_name}`. This command triggers a more exhaustive scan of the contents, to assure an additional TEX run to typeset the label when needed. *(I have no idea how to formulate this sentence.)*

my_macros.mp:

```
vardef label_for_vertex(expr n) =
 textext("$v_" & decimal n & "$")
enddef
```

TEX file:

```
\forceMPTEXcheck{label_for_vertex}

\startMPcode
input my_macros ;
for i=1 upto 6:
 draw label_for_vertex(i) shifted ((1cm,0) rotated 60i);
endfor;
\stopMPcode
```



4 \sometxt{Some really nice features}

4.1 Why is \sometxt so much better than textext?

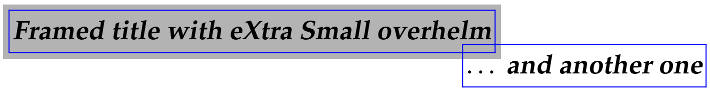
(you may skip this if you want; it's not finished yet anyway)

- There is one obvious reason: **speed**.
  Processing each graphic needs a separate METAPOST run and processing text inside a graphic needs a separate ConTEXt run. Usually the document is processed twice, which means additional four ConTEXt runs plus some conversions just to get those two simple(TODO: use better word!) graphics in the previous section. And all that appears to be extremely slow.
  But there are more reasons than just the efficiency.
- document-wide definitions are seen
  The fact that `textext` triggers a new ConTEXt run doesn't only affect efficiency, but also the scope of your definitions. Inside of `textext` you can only use standard ConTEXt macros since the content is compiled in an isolated environment. Well, there are some exceptions to the rule, but most of them are just calling for troubles. There is but even such simple things as definitions with arguments will fail to work. You can `\input` a file with definitions however, which might be slightly more reliable.
  TODO: EXAMPLE!
- problems with expansion
  TODO: I don't know how to explain it, but as far as I remember math expressions (fractions perhaps) never worked as they were supposed to
- less characters to escape
  TODO: Assign a problem to write a double quote into textext

4.2 Shortcuts

If you need to use the same command inside \sometxt multiple times, you can define a shortcut for it with \definetextext[name]{\command}. This will make \sometxt[name]{whatever} equal to \sometxt{\command{whatever}}.

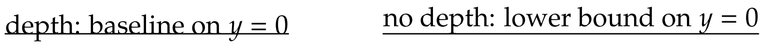
```
\def\ForThoseWhoReallyHateLongCommands#1%
    {\framed[framecolor=blue]{\strut\bs #1}}
\definetextext[xs]{\ForThoseWhoReallyHateLongCommands}

\startMPcode
picture p; p = \sometxt[xs]{Framed title with eXtra Small overhelm};
fill bbox p withcolor .7white;
draw p;
draw \sometxt[xs]{\dots\ and another one} shifted (6cm,-4.5mm);
\stopMPcode
```



4.3 Baseline placement

By default texts are placed on the baseline *(which is OK: I guess that this sectio is not needed).*

```
\startMPcode
vardef place(expr p) =
 image(draw p;
       draw (xpart llcorner p,0)--(xpart lrcorner p,0);)
enddef;

draw place(\sometxt[d]{depth: baseline on $y=0$}) shifted (-5cm,0);
draw place(\sometxt[n]{no depth: lower bound on $y=0$});

\stopMPcode
```



4.4 Accessing text labels from metapost

\sometxt can be used inside inline graphics (for example inside \startMPcode, \startuseMPgraphic etc.), but not in plain .mp files or in some more complex cases. The following example shows how to define text labels and how to access them from metapost.

```
\startTeXtexts
 \TeXtext{3}{Hello}
 \TeXtext{202}{world!}
\stopTeXtexts

\startMPcode
 % this could be written in an external metapost file
 draw sometxt(3) withcolor red;
 draw sometxt(202) shifted (1cm,0) withcolor blue;
\stopMPcode
```

Hello world!

Watch out. \sometxt replaces what has previously been defined with \TeXtext, so the following example is not really a bug. If you want to prevent clashes, use higher numbers in arguments.

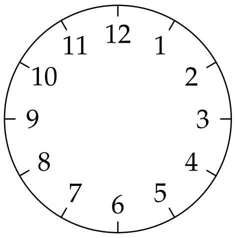
```
\startTeXtexts
 \TeXtext{2}{Hello}
 \TeXtext{1}{world!}
\stopTeXtexts

\startMPcode
 % this could be written in an external metapost file
 draw sometxt(2) withcolor red;
 draw sometxt(1) shifted (1cm,0) withcolor blue;
 draw \sometxt{bug!} shifted (3cm,0);
\stopMPcode
```

Hello bug!   bug!

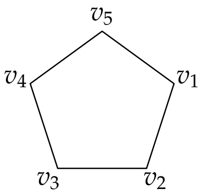
4.5 Loops

Let's see how the "textext" examples can be rewritten using sometxt with predefined labels:

```
\startTeXtexts
 % equivalent to \TeXtext{i}{i}
 \dorecurse{12}{\TeXtext{\recurselevel}{\recurselevel}}
\stopTeXtexts

\startMPcode
 numeric r; r = 1.5cm;
 draw fullcircle scaled 2r;
 for i=1 upto 12:
  draw (origin--down) scaled 4pt shifted (0,r) rotated -30i;
  label(sometxt(i), up scaled .75r rotated -30i);
 endfor;
\stopMPcode
```



```
\startTeXtexts
 % equivalent to \TeXtext{i}{$v_i$}
 \dorecurse{5}{\TeXtext{\recurselevel}{$v_{\recurselevel}$}}
\stopTeXtexts

\startMPcode
 numeric r; r = 1cm;
 pair v[];
 path p;

 for i=1 upto 5:
  % define the five vertices
  v[i] = up scaled r rotated -72i;
  % draw labels: $v_1$, $v_2$, ... $v_5$
  label(sometxt(i), v[i] scaled 1.2);
 endfor;
 % define and draw the pentagon
 p = for i=1 upto 5: v[i]-- endfor cycle; draw p;
\stopMPcode
```



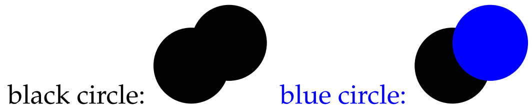
A careful reader will notice a subtle difference in the two graphics: the one created with `textext` uses Latin Modern (default font), while the one created with `\sometxt` uses the document font. Those who want to use the same font with `textext` should use something like this to the document preamble:

```
\startMPenvironment[global]
   \usetypescript[palatino][ec]
   \setupbodyfont[palatino,10pt]
\stopMPenvironment
```

4.6 currentcolor

When no color is specified, metapost uses black by default. If you want to draw something in the same color as the surrounding text, you can use `\MPcolor{currentcolor}`.

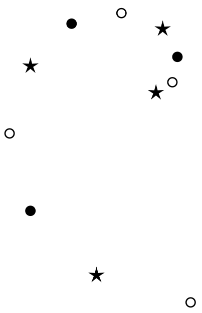
```
\startuseMPgraphic{current color}
 fill fullcircle scaled 1cm shifted (-5mm,-3mm);
 fill fullcircle scaled 1cm withcolor \MPcolor{currentcolor};
\stopuseMPgraphic

\hbox{          black circle: \useMPgraphic{current color}
\color[blue]{    blue circle: \useMPgraphic{current color}}}
```



4.7 More complex example

(Needs more comments.)

```
\defineconversion
    [MySym]
    [$\star$,$\circ$,$\bullet$]

% define the labels
\startTeXtexts
\doloop{
    \doifelseconversionnumber{MySym}{\recurselevel}
        {\TeXtext{\recurselevel}{\convertnumber{MySym}{\recurselevel}}}
        {\exitloop}
}
\stopTeXtexts

% remember the number of different symbols
\doloop{
    \doifelseconversionnumber{MySym}{\recurselevel}
        {\edef\numberofsymbols{\recurselevel}}
        {\exitloop}
}

\startMPcode
for i=0 upto 10:
    draw sometxt(i mod \numberofsymbols + 1)
         shifted ((right scaled 8i) rotated 28i);
endfor;
\stopMPcode
```



5 Feature requests

5.1 Shortcuts with optional parameters

```
\sometxt[my][iwona,20pt]{How can this be typeset with 20pt iwona}
\sometxt[my]{and this with the document font?}

% should become equivalent to
\sometxt{\switchtobodyfont[iwona,20pt]\strut How can this ...}
```

5.2 rename \sometxt to \textext

5.3 make \TeXtextcolormode default to 0

6 Summary

`btex ... etex`	deprecated: inefficient, not flexible	A METAPOST command. `\sometxt` should be used in ConTEXt instead.
`textext("...")`	inefficient, but flexible	A MetaFun command, which enables concatenation of strings and thus **dynamic** generation of labels.
`\texttext{...}`	inefficient, not flexible	This is only a wrapper around `textext(...)`. `\sometxt` should be used instead;
`\sometxt{...}`	efficient, not flexible	The **recommended** command to typeset text, unless dynamic labels are needed.
`sometxt(`*number*`)`	low-level	Used to access labels, previously defined with `\TeXtext{number}{...}`

`\startTeXtexts`
`   \TeXtext{number}{...}`
`\stopTeXtexts`
		For advanced usage only: to define labels that can be drawn later with `sometxt(number)`.