FONT SPECIALS

Bogusław Jackowski
Krzysztof Leszczyński

Kerkrade 2001

cmdfont

What special have we got in Poland?

Kerkrade 2001

What special have we got in Poland?

$\TeX$

Kerkrade 2001

What special have we got in Poland?

TEX

METAFONT

Kerkrade 2001

What special have we got in Poland?

TEX METAPOST METAFONT

Kerkrade 2001

What’s so special about METAPOST specials?

All METAPOST `special` commands migrate to the beginning of the resulting `EPS` file; their order, however, is preserved. TEX `specials`, that might creep into the picture, are simply ignored.

An example of a vexingly hard problem: how to colour selected fragments in texts to be processed by TEX?

A more general problem: how to employ the multitude of existing TEX macro packages, usually special-infested?

SOLUTION

The remedy is to use a special font.

If we could use a particular font in such a way that texts to be typeset using this font would be actually *interpreted* (for example, by external processors) rather than *typeset*, we would circumvent the problem of METAPOST limitations imposed on `special` commands.

`cmdfont`: **what is it?**

```
designsize:=10bp/pt - epsilon;
fontdimen 2: designsize;
fontmaking:=1;

for i:=0 upto 255:
 beginfig(i-256);
  charwd:=charht:=chardp:=charic:=0;
 endfig;
endfor

end.
```

What do typeset texts look like?

- Using `infont` operation:

```
draw "META POST" infont "cmr10";
```

```
...
(META POST) cmr10 9.96265 fshow
...
```

- Using `btex ... etex`: construction

```
draw btex \font\f=cmdfont \f META POST etex;
```

```
...
(META) cmdfont 10 fshow
9.9999 0 moveto
(POST) cmdfont 10 fshow
...
```

How to colour TEX stuff using `cmdfont`

```
verbatimtex
  \def\incmyk#1#2{%
   \leavevmode\rlap{\font\f=cmdfont \f
     gsave #1 setcmykcolor}%
   #2\hbox{\font\f=cmdfont \f grestore}}
etex
beginfig(100);
 draw btex \vbox{
  \hsize 40mm \pretolerance10000
   \raggedright \noindent
   A simple method to tint a \incmyk{1 0 1 0}
   {{\bf selected fragment}} of a text.} etex;
endfig;
```

The result of colouring:

A simple method to tint a **selected fragment** of a text.

It would be nice to be able to include virtually any TEX files into METAPOST!

We can achieve this by redefining every `\special` into a macro that produces a zero-sized `\hbox` containing the “special,” i.e., a text typeset using `cmdfont`. The pictures (POSTSCRIPT files) need to be postprocessed in order to convert the resulting strings back into appropriate POSTSCRIPT fragments.

Some `\special` commands are difficult to interpret in METAPOST applications

‘`papersize`’ specials should not occur in `EPS` files; ignoring them is probably the best thing we can do.

‘`ps::`’ specials (note the double colons) are somewhat obscurely documented. Roughly, they insert literal POSTSCRIPT. Feel warned that they might lead into troubles.

‘`!`’ specials (header POSTSCRIPT fragments) need to be put in the initialisation part of the `EPS`. The simplest solution is the restitution of the original `\special` commands by putting them into and auxiliary TEX file.

External processing

A perl script `despecial` converts the specials in the final PostScript file into a PostScript code and auxiliary TeX files according to a defined set of rules.

Rules:

- precedence – a real number
- a matching routine, usually a regular expression
- a service routine

```
despecial_rule( 0, {/^ps::/},
   process_ps_colon_colon );

despecial_rule( 0.1, {/^ps:/},
   process_ps_colon );

despecial_rule( 0,
   {/^despecial:\s+(.*)$/s},
   {eval "package despecial;$1"});
```

Each rule is matched until the one with the lowest precedence is found. If there are 2 or more matching routines with the same precedence a warning is issued.

Perl-in-TEX code

despecial can be fed from TEX or METAPOST script using the despecial: prefix. You can even define your own rules.

```
\special{despecial:
  print STDERR "Hello World\\n")}

\special{despecial:
  sub skip_code {
    $save_line_number = \$\#current_file}
  sub end_skip_code {
    \$\#current_file = $save_line_number}
  add_rule(0, {/^skip_code $/}, my_special)
  add_rule(0, {/^end_skip_code$/},
      end_skip_code )}
```

Kerkrade 2001

Very important is the ‘`psfile`’ special (used, e.g., by `epsf.tex`)

It has many parameters, namely `llx`, `lly`, `urx`, `ury`, used for bounding box. It’s tempting to use PostScript functions `@llx`, `@lly`, `@urx`, and `@ury` provided by `dvips`.

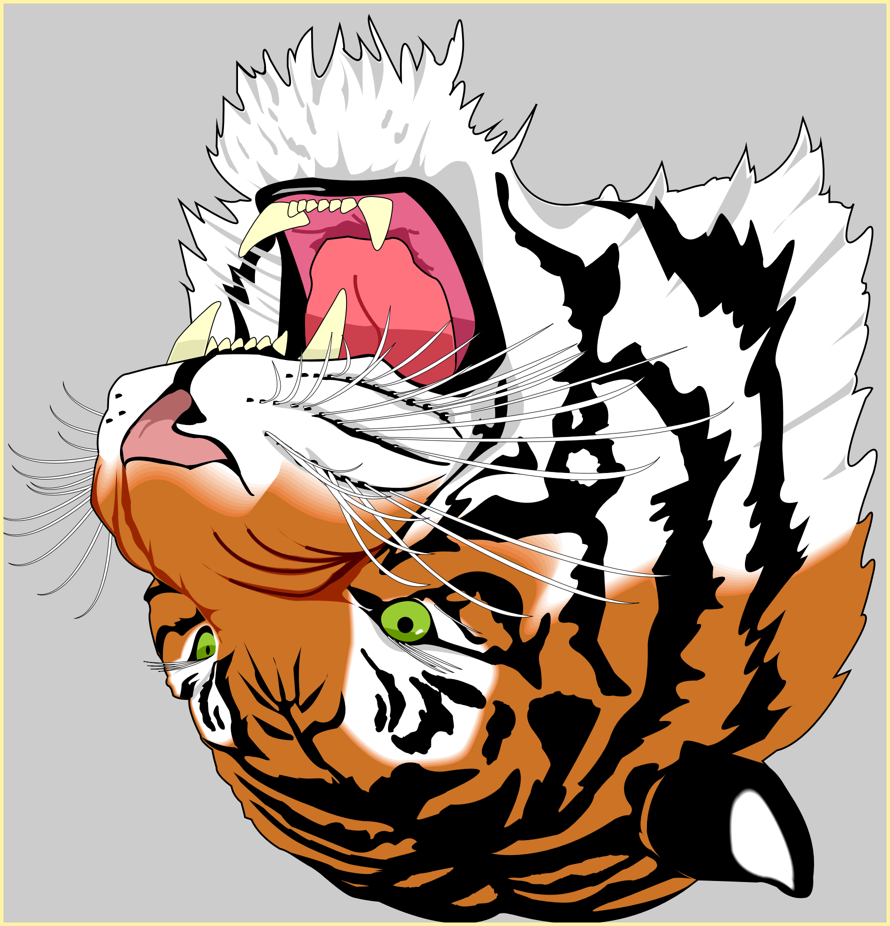
The following `\special`:

```
\special{PSfile=tiger.ps llx=22 lly=171
         urx=567 ury=738 rwi=283 clip}
```

one would intuitively convert to:

```
@beginspecial
 22 @llx 171 @lly 567 @urx 738 @ury
 283 @rwi
@clip
@setspecial
```

… which, however, may be considered to be wrong:



Kerkrade 2001

Summary

What do we know about `cmdfont` on Wednesday, 26th September, 11:15?

We think we can:

- put our own PostScript fragments into proper places in METAPOST output files
- extend the set of PostScript operators produced by METAPOST (`eofill`, `eoclip`)
- enrich `btex ... etex` constructions (e.g., coloring)
- use virtually any TEX files as METAPOST `picture` objects
- insert mark objects into any TEX or METAPOST file for postprocessing

That's it.

Using `cmdfont` opens a broad road leading to a new realm of pre-, sub- and postprocessing applications. Frankly, we are not aware of most of them. One should be warned, however, about the existence of many reefs and rocky islands with bouncing tigers on them.

Kerkrade 2001

That's it.