Selection in PDF viewers and a LuaTEX bug

Hans Hagen

In January 2014 a message was posted to the ConTEXt mailing list asking for clarification about the way PDF viewers select text. Let me give an example of that (inside a convenient ConTEXt wrapper):

```
\startTEXpage[offset=2cm]
  \hbox{$ x+y $}
\stopTEXpage
```

In figure 1 you can see how for instance Acrobat (which I use for proofing) and SumatraPDF (which I use in my edit–preview cycle) select this text. As reported in the mail, other viewers behave like SumatraPDF does, with excessive vertical space included in the selection.

Figure 1: Some math selected in PDF viewers.

Part of the question was why wrapping a display formula in an `\hbox` doesn't have this effect on viewers. Think of:

```
\startTEXpage[offset=2cm]
  \hbox{\startformula x+y \stopformula}
\stopTEXpage
```

This can be reduced to the following primitive construct (which is rendered in figure 2):

```
\startTEXpage[offset=2cm]
  \hbox{$$ x+y $$}
\stopTEXpage
```

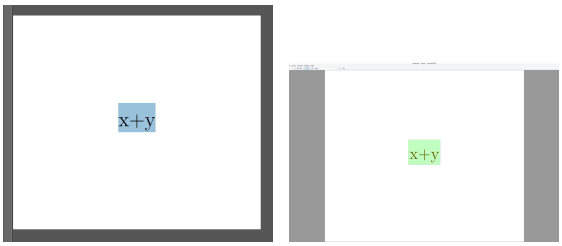

Figure 2: Some text selected in PDF viewers.

If you look closely you will see that we have text and not math. This is because in restricted horizontal mode (inside `\hbox`) TEX sees the `$$` as a begin and immediate end of math mode, so in fact we have here some text surrounded by empty math. When I realized that using ConTEXt's `\startformula` could have this side effect in some situations I decided to catch this, but sometimes TEX can give surprises.

Already a while ago Taco and I decided that it would be handy to have primitives in LuaTEX for special characters (or more precisely: characters with certain catcodes) as used in math and alignments.

Table 1: Some of the extra LuaTEX primitives.

token	primitive(s)	
#	\alignmark	
&	\aligntab	
$	\Ustartmath	\Ustopmath
$$	\Ustartdisplaymath	\Ustopdisplaymath
^	\Usuperscript	
_	\Usubscript	

As you can see in table 1, the dollars are somewhat special as in fact we don't alias characters (tokens) but have introduced primitives that change the mode from text to inline or display math and back. So, we can say:

```
\startTEXpage[offset=2cm]
 \hbox{\Ustartdisplaymath x+y \Ustopdisplaymath}
\stopTEXpage
```

This renders okay in LuaTEX 0.78.1, but when we tinker a bit like this (with an invalid `\par` inside the `\hbox`):

```
\startTEXpage[offset=2cm]
  \hbox{\Ustartdisplaymath x+y \Ustopdisplaymath
        \par}
\stopTEXpage
```

we get this:

```
Assertion failed!

Program: c:\tex\texmf-win64\bin\luatex.exe
File: web2c/luatexdir/tex/texnodes.w, Line 830

Expression: p> my_prealloc
```

Oops. Furthermore, if we add some text after the invalid `\par`:

```
\startTEXpage[offset=2cm]
  \hbox{\Ustartdisplaymath x+y \Ustopdisplaymath
        \par x}
\stopTEXpage
```

we get:

```
! This can't happen (vpack).
```

As an excursion from working on the Critical Editions project Luigi Scarso and I immediately

started debugging this at the engine level and after some tracing we saw that it had to do with packaging. Taco joined in, and we decided that it made no sense at all to try to deal with this at that level simply because we ourselves had bypassed a natural boundary of TEX: caching the start of display math by seeing two successive $ as an inline formula. So the solution is either to make \Ustartdisplaymath more clever, but better is to simply issue an error message when this state is entered. That way we stick to the original TEX point of view, an approach that has never failed us so far. The chosen solution is to issue error messages (broken onto two lines for *TUGboat*):

```
! You can't use `\Ustartdisplaymath'
  in restricted horizontal mode

! You can't use `\Ustopdisplaymath'
  in restricted horizontal mode
```

If you really want this you can redefine the primitive:

```
\let\normalUstartmath\Ustartmath
\let\normalUstopmath \Ustopmath

\let\normalUstartdisplaymath\Ustartdisplaymath
\let\normalUstopdisplaymath \Ustopdisplaymath

\unexpanded\def\Ustartdisplaymath % context way
 {\ifinner
   \ifhmode
     \normalUstartmath
     \let\Ustopdisplaymath
         \normalUstopmath
   \else
     \normalUstartdisplaymath
     \let\Ustopdisplaymath
         \normalUstopdisplaymath
   \fi
  \else
    \normalUstartdisplaymath
    \let\Ustopdisplaymath
        \normalUstopdisplaymath
  \fi}
```

As with many things in TEX there is often a way out, as long as things are open and accessible enough. Currently in ConTEXt we do something like the above for cases where confusion can happen.

With that fixed, it was time to return to the original question. Why do math selections have such large bounding boxes in some viewers? The answer to that is in the PDF file. Let's look at the font properties of a math font, Latin Modern Math here:

```
24 0 obj
  <<
    /Type        /FontDescriptor
    /FontName    /CXDZIF+LatinModernMath-Regular
    /Flags       4
    /FontBBox    [-1042 -3060 4082 3560]
    /Ascent      3560
    /CapHeight   683
    /Descent     -3060
    /ItalicAngle 0
    /StemV       93
    /XHeight     431
    /FontFile3   23 0 R
    /CIDSet      22 0 R
  >>
endobj
```

Compare these rather large values for FontBBox, Ascent, etc., with a text font (Latin Modern Roman):

```
24 0 obj
  <<
    /Type        /FontDescriptor
    /FontName    /TEKCPF+LMRoman12-Regular
    /Flags       4
    /FontBBox    [-422 -280 1394 1127]
    /Ascent      1127
    /CapHeight   683
    /Descent     -280
    /ItalicAngle 0
    /StemV       91
    /XHeight     431
    /FontFile3   23 0 R
    /CIDSet      22 0 R
  >>
endobj
```

So, it looks like Acrobat is using the actual heights and depths of glyphs (probably with some slack) while other viewers use the font's ascender and descender values. So in the end the answer is: there is nothing the user, ConTEXt or LuaTEX can do about it, apart from messing with the values above, which is probably not a good idea.

But trying to answer the question (by stripping down, etc.) had the side effect of identifying a bug in LuaTEX. A lesson learned is thus that even adding simple primitives like the ones above needs some studying of the source code in order to identify side effects. We should have known!

◇ Hans Hagen
Pragma ADE
http://pragma-ade.com