MPlib

1

ConTEXt

2

MkIV

3

We started using MetaPost over ten years ago. Graphics were embedded as eps.

1

We also added some extensions (using specials) like shading, transparency and support for processcolors, spotcolors and multitones.

3

By now the mechanisms are pretty stable and frequently used by users. No real in-depth knowledge is needed.

9

Reusing graphics has always been part of the game, either or not based on the current state of the document (dimensions, colors, etc.).

6

Embedding text was taken care of as efficient as possible, later by using some trickery that avoided separate runs altogether.

5

Challenged by Sebastian Rahtz we wrote a MetaPost to pdf converter in TEX so that we could use them directly.

2

Graphics know a bit about the current situation, i.e. layout, font and other dimensions are passed along.

8

At some point a mechanism appeared to include MetaPost code in the document source. Processing could take place between runs or directly (write18).

4

Such graphics are rather well integrated in background mechanisms and can adapt themselves to situations.

7

In education documents we often need backgrounds, rules in margins, underlining and special makeup of section and chapter titles.

1

This may add several seconds or runtime per page and more when we deal with text in MetaPost (which can be avoided).

3

Using a tight integration (i.e. a library) made more sense and therefore the mplib project was started.

8

This easily mounts up to tens of graphics per page, even when graphic data is collected.

2

The expectation was that by staying inside TEX we could gain a lot. Of course MetaPost still had to do some work.

5

In ConTEXt MkIV we already had reimplemented the MetaPost to pdf converter which in Lua is a bit faster than in TEX (the bottleneck is now in the literals).

6

That library would focus on the graphic part as it was expected that text could be dealt with at the TEX end.

9

Some experiments (with Fabrice) demonstrated that using pipes was too fragile in the current situation (timing problems).

7

For special purposes like flowcharts and gnuplot graphics runtime may even be more influenced by calling MetaPost.

4

All that users now see of MetaPost is the reported runtime and of course error messages (these go to the TEX log).

8

Experiments with the first version of the library showed that we could easily get a throughput of thousands of graphics per second (processing and conversion).

3

4

Multiple runs for a graphic (as used for special text processing and outlines) is handled by MkIV internally in such a way that processing time is hardly influenced.

7

Although we could have used the PostScript parser, it made more sense to operate on the raw output (represented in tables).

2

We had expected to be able to use the relative new pre/postscript features of MetaPost, but this mechanism needs to be extended in order to replace all special based tricks.

6

In everyday documents MetaPost runtime has become close to zero, and in complex documents neglectable compared to the overall runtime.

9

The most complex part was (as usual) dealing with paths drawn by special pens, a complication that eventually resulted in a proper helper function.

5

In order to test the library the MetaPost to pdf converter had to be rewritten (again).

1

All existing mechanisms are supported in ConTEXt MkIV. It really helps that users are eager to update and test.

1

Document styles that operate close to what is reasonable now behave rather normal. We currently test these mechanisms on real projects.

8

Multiple formats are supported but not yet at the user interface level. Soon each graphic can get a format attached.

3

Tight integration of MetaPost resulted in many users using these features. We expect even more usage due to the neglectable runtime.

9

In our reference document of (currently) 240 pages the 66 graphics take .35 seconds. The speed gain is even more noticeable for the LuaTEX manual.

7

We will also support multiple instances of a format so that user graphics will not interfere with system graphics (this is handy for modules).

4

We will use mplib for runtime font generation. Tests show that a generation speed of 500-1000 glyphs with pens per second uncached is feasible (Dell M90 with Vista).

5

Eventually mplib might produce proper charstrings that then can be used to construct (and extend) real fonts on the fly.

6

MetaPost format generation is done automatically and are kept in a ConTEXt specific namespace (bound to the TEX format).

2