

The luamplib package

Hans Hagen, Taco Hoekwater, Elie Roux, Philipp Gesang and Kim Dohyun

Current Maintainer: Kim Dohyun

Support: <https://github.com/lualatex/luamplib>

2025/12/05 V2.37.7

Abstract

Package to have METAPOST code typeset directly in a document with Lua \TeX

Contents

1	Documentation	2
1.1	\TeX	3
1.1.1	<code>\mplibforcehmode</code>	3
1.1.2	<code>\everymplib, \everyendmplib</code>	3
1.1.3	<code>\mplibsetformat</code>	3
1.1.4	<code>\mplibnumbersystem</code>	4
1.1.5	<code>\mplibshowlog</code>	4
1.1.6	<code>\mpliblegacybehavior</code>	4
1.1.7	<code>\mplibtexttextlabel</code>	5
1.1.8	<code>\mplibcodeinherit</code>	5
1.1.9	<code>\mplibglobaltexttext</code>	6
1.1.10	Separate METAPOST instances	6
1.1.11	<code>\mplibverbatim</code>	7
1.1.12	<code>\mpdim</code>	7
1.1.13	<code>\mpcolor</code>	7
1.1.14	<code>\mpfig, \endmpfig</code>	7
1.1.15	About cache files	8
1.1.16	About figure box metric	8
1.1.17	<code>luamplib.cfg</code>	8
1.1.18	Tagged PDF	9
1.2	METAPOST	10
1.2.1	<code>mplibdimen, mplibcolor</code>	10
1.2.2	<code>mplibtexcolor, mplibrbgtexcolor</code>	11
1.2.3	<code>mplibgraphictext</code>	11
1.2.4	<code>mplibglyph</code>	11

1.2.5	<code>mplibdrawglyph</code>	12
1.2.6	<code>mpliboutlinetext</code>	12
1.2.7	<code>\mppattern</code> , <code>\endmppattern</code> , <code>withmppattern</code>	12
1.2.8	<code>withfademethod</code>	15
1.2.9	<code>asgroup</code>	16
1.2.10	<code>\mplibgroup</code> , <code>\endmplibgroup</code>	17
1.2.11	<code>withtransparency</code>	18
1.2.12	<code>withshadingmethod</code>	18
1.2.13	<code>mpliblength</code> , <code>mplibuclength</code>	19
1.2.14	<code>mplibsubstring</code> , <code>mplibucsubstring</code>	20
1.3	<code>Lua</code>	20
1.3.1	<code>runscript</code>	20
1.3.2	<code>luamplib.instances</code>	20
1.3.3	<code>luamplib.process_mplibcode</code>	21
2	Implementation	21
2.1	<code>Lua module</code>	21
2.2	<code>TeXpackage</code>	88
3	The GNU GPL License v2	108

1 Documentation

This package aims at providing a simple way to typeset directly METAPOST code in a document with Lua_{TeX}. Lua_{TeX} is built with the Lua `mplib` library, that runs METAPOST code. This package is basically a wrapper for the Lua `mplib` functions and some `TeX` functions to have the output of the `mplib` functions in the pdf.

Using this package is easy: in Plain, type your METAPOST code between the macros `\mplibcode` and `\endmplibcode`, and in `LaTeX` in the `mplibcode` environment.

The resulting METAPOST figures are put in a `TeX` hbox with dimensions adjusted to the METAPOST code.

The code of `luamplib` is basically from the `luatex-mplib.lua` and `luatex-mplib.tex` files from Con`TeX`t. They have been adapted to `LaTeX` and Plain by Elie Roux and Philipp Gesang and new functionalities have been added by Kim Dohyun. The most notable changes are:

- possibility to use `btex ... etex` to typeset `TeX` code. `texttext <string>` is a more versatile macro equivalent to `TEX <string>` from `TEX.mp`. `TEX` is also allowed and is a synonym of `texttext`. The argument of `mplib`'s primitive `maketext` will also be processed by the same routine.
- possibility to use `verbatimtex ... etex`, though it's behavior cannot be the same as the stand-alone `mpost`. Of course you cannot include `\documentclass`, `\usepackage` etc. When these `TeX` commands are found in `verbatimtex ... etex`, the entire code will be ignored. The treatment of `verbatimtex` command has changed a lot since v2.20: see below 1.1.6.

- in the past, the package required PDF mode in order to have some output. Starting with version 2.7 it works in DVI mode as well, though DVIPDFMx is the only DVI tool currently supported.

It seems to be convenient to divide the explanations of some more changes and cautions into three parts: T_EX, METAPOST, and Lua interfaces.

1.1 T_EX

1.1.1 `\mplibforcehmode`

When this macro is declared, every METAPOST figure box will be typeset in horizontal mode, so `\centering`, `\raggedleft` etc will have effects. `\mplibnoforcehmode`, being default, reverts this setting.¹

1.1.2 `\everymplib{...}`, `\everyendmplib{...}`

`\everymplib` and `\everyendmplib` redefine the lua table containing METAPOST code which will be automatically inserted at the beginning and ending of each METAPOST code chunk.

```
\everymplib{ beginfig(0); }
\everyendmplib{ endfig; }
\begin{mplibcode}
  % beginfig/endfig not needed
  draw fullcircle scaled 1cm;
\end{mplibcode}
```

1.1.3 `\mplibsetformat{plain|metafun}`

There are (basically) two formats for METAPOST: *plain* and *metafun*. By default, the *plain* format is used, but you can set the format to be used by future figures at any time using `\mplibsetformat{<format name>}`.

N.B. As *metafun* is such a complicated format, we cannot support all the functionalities producing special effects provided by *metafun*. At least, however, transparency (actually opacity), shading (gradient colors) and transparency group are fully supported, and `outlinetext` is supported by our own alternative `mpliboutlinetext` (see below 1.2.6). You can try other effects as well, though we did not fully tested their proper functioning.

transparency (texdoc metafun §8.2) Transparency is so simple that you can apply it to an object, with *plain* format as well as *metafun*, just by appending withprescript `"tr_transparency=<number>"` to the sentence. ($0 \leq \langle number \rangle \leq 1$)

From v2.36, `withtransparency` is available with *plain* as well. See below 1.2.11.

¹Actually these commands redefine `\prependtomplibox`. So you can redefine this command with anything suitable before a box. But see 1.1.18 on Tagged PDF.

shading (texdoc metafun § 8.3) One thing worth mentioning about shading is: when a color expression is given in string type, it is regarded by `luamplib` as a color expression of \TeX side. For instance, when `withshadecolors("orange", 2/3red)` is given, the first color "orange" will be interpreted as a color, `xcolor` or `l3color`'s expression.

From v2.36, shading is available with *plain* format as well with extended functionality. See below [1.2.12](#).

transparency group (texdoc metafun § 8.8) As for transparency group, the current *metafun* document is not correct. The true syntax is:

```
draw <picture>|<path> asgroup <string>
```

where *<string>* should be "" (empty), "isolated", "knockout", or "isolated,knockout". Beware that currently many of the PDF rendering applications, except Adobe Acrobat, cannot properly render the isolated or knockout effect.

Transparency group is available with *plain* format as well, with extended functionality. See below [1.2.9](#).

1.1.4 `\mplibnumbersystem{scaled|double|decimal}`

Users can choose `numbersystem` option. The default value is `scaled`, which can be changed by declaring `\mplibnumbersystem{double}` or `\mplibnumbersystem{decimal}`.

1.1.5 `\mplibshowlog{enable|disable}`

Default: `disable`. When `\mplibshowlog{enable}`² is declared, log messages returned by the `METAPOST` process will be printed to the `.log` file. This is the \TeX side interface for `luamplib.showlog`.

1.1.6 `\mpliblegacybehavior{enable|disable}`

By default, `\mpliblegacybehavior{enable}` is already declared for backward compatibility, in which case \TeX code in `verbatimtex ... etex` that comes just before `beginfig()` will be inserted before the following `METAPOST` figure box. In this way, each figure box can be freely moved horizontally or vertically. Also, a box number can be assigned to a figure box, allowing it to be reused later.³

```
\mplibcode
verbatimtex \moveright 3cm etex; beginfig(0); ... endfig;
verbatimtex \leavevmode etex; beginfig(1); ... endfig;
verbatimtex \leavevmode\lower 1ex etex; beginfig(2); ... endfig;
verbatimtex \endgraf\moveright 1cm etex; beginfig(3); ... endfig;
\endmplibcode
```

²As for user's setting, `enable`, `true` and `yes` are identical; `disable`, `false` and `no` are identical.

³But the recommended way to reuse a figure is using `\mplibgroup` command. See below [1.2.10](#).

N.B. `\endgraf` should be used instead of `\par` inside `verbatimtex ... etex`.

On the other hand, \TeX code in `verbatimtex ... etex` between `beginfig()` and `endfig` will be inserted after flushing out the `METAPOST` figure. As shown in the example below, `VerbatimTeX` (*string*) is a synonym of `verbatimtex ... etex`.

```
\mplibcode
D := sqrt(2)**7;
beginfig(0);
draw fullcircle scaled D;
VerbatimTeX("\gdef\Dia{" & decimal D & "}");
endfig;
\endmplibcode
diameter: \Dia bp.
```

By contrast, when `\mpliblegacybehavior{disable}` is declared, any `verbatimtex ... etex` will be executed, along with `btex ... etex`, sequentially one by one. So, some \TeX code in `verbatimtex ... etex` will have effects on following `btex ... etex` codes.

```
\begin{mplibcode}
beginfig(0);
draw btex ABC etex;
verbatimtex \bfseries etex;
draw btex DEF etex shifted (1cm,0); % bold face
draw btex GHI etex shifted (2cm,0); % bold face
endfig;
\end{mplibcode}
```

1.1.7 `\mplibtexttextlabel{enable|disable}`

Default: `disable`. `\mplibtexttextlabel{enable}` enables the labels typeset via `texttext` instead of `infont` operator. So, `label("my text", origin)` thereafter is exactly the same as `label(texttext "my text", origin)`.

N.B. In the background, `luamplib` redefines `infont` operator so that the right side argument (the font part) is totally ignored. Therefore the left side argument (the text part) will be typeset with the current \TeX font.

From v2.35, however, the redefinition of `infont` operator has been revised: when the character code of the text argument is less than 32 (control characters), or is equal to 35 (#), 36 (\$), 37 (%), 38 (&), 92 (\), 94 (^), 95 (_), 123 ({), 125 (}), 126 (~) or 127 (DEL), the original `infont` operator will be used instead of `texttext` operator so that the font part will be honored. Despite the revision, please take care of `char` operator in the text argument, as this might bring unpermitted characters into \TeX .

1.1.8 `\mplibcodeinherit{enable|disable}`

Default: `disable`. `\mplibcodeinherit{enable}` enables the inheritance of variables, constants, and macros defined by previous `METAPOST` code chunks. On the contrary, `\mplibcodeinherit{disable}` will make each code chunk being treated as an independent instance, never affected by previous code chunks.

1.1.9 `\mplibglobaltexttext{enable|disable}`

Default: `disable`. Formerly, to inherit `btex ... etex` boxes as well as other METAPOST macros, variables and constants, it was necessary to declare `\mplibglobaltexttext{enable}` in advance. But from v2.27, this is implicitly enabled when `\mplibcodeinherit` is enabled. This optional command still remains mostly for backward compatibility.

```
\mplibcodeinherit{enable}
%\mplibglobaltexttext{enable}
\everymplib{ \beginfig(0);} \everyendmplib{ \endfig;}
\mplibcode
  label(btex  $\sqrt{2}$  etex, origin);
  draw fullcircle scaled 20;
  picture pic; pic := currentpicture;
\endmplibcode
\mplibcode
  currentpicture := pic scaled 2;
\endmplibcode
```

1.1.10 Separate METAPOST instances

`luamplib` v2.22 has added the support for several named METAPOST instances in \TeX `mplibcode` environment. Plain \TeX users also can use this functionality. The syntax for \TeX is:

```
\begin{mplibcode}[instanceName]
  % some mp code
\end{mplibcode}
```

The behavior is as follows.

- All the variables and functions are shared only among all the environments belonging to the same instance.
- `\mplibcodeinherit` only affects environments with no instance name set (since if a name is set, the code is intended to be reused at some point).
- `btex ... etex` boxes are also shared and do not require `\mplibglobaltexttext`.
- When an instance names is set, respective `\currentmpinstancename` is set as well.

In parallel with this functionality, we support optional argument of instance name for `\everymplib` and `\everyendmplib`, affecting only those `mplibcode` environments of the same name. Unnamed `\everymplib` affects not only those instances with no name, but also those with name but with no corresponding `\everymplib`. The syntax is:

```
\everymplib[instanceName]{...}
\everyendmplib[instanceName]{...}
```

1.1.11 `\mplibverbatim{enable|disable}`

Default: `disable`. Users can issue `\mplibverbatim{enable}`, after which the contents of `mplibcode` environment will be read verbatim. As a result, except for `\mpdim` and `\mpcolor` (see 1.1.12 and 1.1.13), all other \TeX commands outside of the `btex` or `verbatimtex ... etex` are not expanded and will be fed literally to the `mplib` library.

1.1.12 `\mpdim{...}`

Besides other \TeX commands, `\mpdim` is specially allowed in the `mplibcode` environment. This feature is inspired by `gmp` package authored by Enrico Gregorio. Please refer to the manual of `gmp` package for details.

```
\begin{mplibcode}
  beginfig(1)
    draw origin--(.6\mpdim{\linewidth},0) withpen pencircle scaled 4
    dashed evenly scaled 4 withcolor \mpcolor{orange};
  endfig;
\end{mplibcode}
```

1.1.13 `\mpcolor[...]{...}`

With `\mpcolor` command, color names or expressions of `color`, `xcolor` and `l3color` module/packages can be used in the `mplibcode` environment (after `withcolor` operator). See the example above at 1.1.12. The optional `[...]` denotes the option of `xcolor`'s `\color` command. For spot colors, `l3color` (in PDF/DVI mode), `colorspace`, `spotcolor` (in PDF mode) and `xespotcolor` (in DVI mode) packages are supported as well.

1.1.14 `\mpfig ... \endmpfig`

Besides the `mplibcode` environment (for $\mathbb{E}\TeX$) and `\mplibcode ... \endmplibcode` (for Plain), we also provide unexpandable \TeX macros `\mpfig ... \endmpfig` and its starred version `\mpfig* ... \endmpfig` to save typing toil. The former is roughly the same as follows:

```
\begin{mplibcode}[@mpfig]
  beginfig(0)
  token list declared by \everymplib[@mpfig]
  ...
  token list declared by \everyendmplib[@mpfig]
endfig;
\end{mplibcode}
```

and the starred version is roughly the same as follows:

```
\begin{mplibcode}[@mpfig]
  ...
\end{mplibcode}
```

In these macros `\mpliblegacybehavior{disable}` is forcibly declared. Again, as both share the same instance name, METAPOST codes are inherited among them. A simple example:

```
\everymplib[@mpfig]{ drawoptions(withcolor .5[red,white]); }
\mpfig* input boxes \endmpfig
\mpfig
  circleit.a(btex Box 1 etex); drawboxed(a);
\endmpfig
```

The instance name (default: `@mpfig`) can be changed by redefining `\mpfiginstancename`, after which a new `mplib` instance will start and code inheritance too will begin anew. `\let \mpfiginstancename\empty` will prevent code inheritance if `\mplibcodeinherit{true}` is not declared.

1.1.15 About cache files

To support `btex ... etex` in external `.mp` files, `luamplib` inspects the content of each and every `.mp` file and makes caches if necessary before returning their paths to the `mplib` library. This could waste the compilation time, as most `.mp` files do not contain `btex ... etex` commands. So `luamplib` provides macros as follows, so that users can give instructions about files that do not require this functionality.

- `\mplibmakenocache{⟨filename⟩[,⟨filename⟩,...]}`
- `\mplibcancelnocache{⟨filename⟩[,⟨filename⟩,...]}`

where `⟨filename⟩` is a filename excluding `.mp` extension. Note that `.mp` files under `$TEXMFMAIN/metapost/base` and `$TEXMFMAIN/metapost/context/base` are already registered by default.

By default, cache files will be stored in `$TEXMFVAR/luamplib_cache` or, if it's not available (mostly not writable), in the directory where output files are saved: to be specific, `$TEXMF_OUTPUT_DIRECTORY/luamplib_cache`, `./luamplib_cache`, `$TEXMFOUTPUT/luamplib_cache`, and `.`, in this order. `$TEXMF_OUTPUT_DIRECTORY` is normally the value of `--output-directory` command-line option.

Users can change this behavior by the command `\mplibcachedir{⟨directory path⟩}`, where tilde (`~`) is interpreted as the user's home directory (on a windows machine as well). As backslashes (`\`) should be escaped by users, it would be easier to use slashes (`/`) instead.

1.1.16 About figure box metric

Notice that, after each figure is processed, the macro `\MPwidth` stores the width value of the latest figure; `\MPheight`, the height value. Incidentally, also note that `\MPllx`, `\MPlly`, `\MPurx`, and `\MPury` store the bounding box information of the latest figure without the unit `bp`.

1.1.17 luamplib.cfg

At the end of package loading, `luamplib` searches `luamplib.cfg` and, if found, reads the file in automatically. Frequently used settings such as `\everymplib`, `\mplibforcehmode` or `\mplibcodeinherit` are suitable for going into this file.

1.1.18 Tagged PDF

When `tagpdf` package is loaded and activated, `mplibcode` environment accepts additional options for tagged PDF. The code related to this functionality is currently in experimental stage, not guaranteeing backward compatibility. Available optional keys are similar to those of the \LaTeX 's `picture` environment (`texdoc latex-lab-graphic`). The default tagging mode is the `alt` key with Figure structure.

alt= $\langle text \rangle$ starts a Figure tag by default and sets an alternate text of the figure from the $\langle text \rangle$. BBox info will be added automatically to the PDF. This key is needed for ordinary `METAPOST` figures, for which, if no alt text is given, a default text will be used with a warning issued. You can change the alternate text within `METAPOST` code as well: `VerbatimTeX "\mplibalttext{\langle text \rangle}";`

actualtext= $\langle text \rangle$ starts a Span tag implicitly and sets a replacement text (a.k.a. actual text) from the $\langle text \rangle$. If in vertical mode, horizontal mode will be forced by `\noindent` command.⁴ BBox info will not be added. This key is intended for figures which can be represented by a character or a small sequence of characters. You can change the actual text within `METAPOST` code as well: `VerbatimTeX "\mplibactualtext{\langle text \rangle}";`

artifact starts an Artifact MC (marked content). BBox info will not be added. This key is intended for decorative figures which have no semantic meaning.

text starts an Artifact MC but enables tagging on \TeX -text boxes (such as `btex ... etex`, excluding pictures made by `infont` operator). If in vertical mode, horizontal mode will be forced by `\noindent` command.⁵ BBox info will not be added. This key is intended for figures the meaning of which is the sequence of texts in the \TeX -text boxes in the order they are drawn in the figure. Inside text-mode figures, reusing \TeX -text boxes is strongly discouraged.

Note that the text in a \TeX -text box which starts with `[taggingoff]` will not be tagged at all, and of course `[taggingoff]` and its trailing spaces will be gobbled by `luamplib`. For example, the first and the third boxes in the following figure will not be tagged, and still remain in the Artifact MC-chunks.

```
\begin{mplibcode}[text]
  beginfig(1)
    draw btex [taggingoff]  $\sqrt{2}$  etex ;
    draw texttext " $\sqrt{3}$ " shifted 10down ;
    draw TEX "[taggingoff]  $\sqrt{5}$ " shifted 20down ;
    draw maketext " $\sqrt{7}$ " shifted 30down ;
    draw mplibgraphictext " $\sqrt{x}$ " shifted 40down ;
  endfig;
\end{mplibcode}
```

⁴It is not recommended to personally redefine `\prependtomplibbox`. Apart from using `\mplibforcehmode` or `\mplibnoforcehmode`, the redefinition might be incompatible with `actualtext` key. See 1.1.1 on these commands.

⁵The key `text` also shares the limitation mentioned in the previous footnote.

off Given this key, nothing will be tagged by `luamplib`.

tag= $\langle name \rangle$ You can choose a tag name, default value being `Figure`.⁶ For instance, you can set `tag=Formula, alt=` $\langle text \rangle$ to get a `Formula` element with its alternate text.⁷

adjust-BBox= $\langle dimens \rangle$ You can correct the `BBox` attribute of the figure by space-separated four dimensional values, which will be added to the automatically calculated `BBox` values. To draw the bounding box for checking with half-transparent red color, you can add `debug=BBox` to the argument of `\DocumentMetadata` command.

tagging-setup= $\langle key-val list \rangle$ This key accepts as its value the list of key-value options mentioned so far.

You can set these tagging options anywhere in the document by declaring `\SetKeys[luamplib/tagging]{` $\langle key-val list \rangle$, which will affect `luamplib` figures thereafter in the scope. And these options are provided also for `\mpfig` and `\usemplibgroup` (see [below](#)) commands.

```
\begin{mplibcode}[myInstanceName, alt=drawing of a circle]
...
\end{mplibcode}

\mpfig[alt=drawing of a square box]
...
\endmpfig

\usemplibgroup[alt=drawing of a triangle]{...}

\mppattern{...}           % see below
  \mpfig[off]             % do not tag this figure
...
\endmpfig
\endmppattern
```

As for the instance name of `mplibcode` environment, `instance=` $\langle name \rangle$ or `instancename=` $\langle name \rangle$ is also allowed in addition to the raw instance name as shown above.

1.2 METAPOST

1.2.1 `mplibdimen ...`, `mplibcolor ...`

These are METAPOST interfaces for the \TeX commands `\mpdim` and `\mpcolor` (see above [1.1.12](#) and [1.1.13](#)). For example, `mplibdimen "\linewidth"` is basically the same as `\mpdim{\linewidth}`, and `mplibcolor "red!50"` is basically the same as `\mpcolor{red!50}`. The difference is that these METAPOST operators can also be used in external `.mp` files, which cannot have \TeX commands outside of the `btex` or `verbatimtex ... etex`.

⁶The option `tag=false`, however, is a synonym of the `off` key.

⁷Beware that this bypasses \LaTeX 's regular math formula tagging, for which the `text` key is needed.

1.2.2 `mplibtexcolor ...`, `mplibrgbtexcolor ...`

`mplibtexcolor`, which accepts a string argument, is a METAPOST operator that converts a T_EX color expression to a METAPOST color expression, that can be used anywhere color expression is expected as well as after the `withcolor` operator. For instance:

```
color col;
col := mplibtexcolor "olive!50";
```

But the result may vary in its color model (gray/rgb/cmyk) according to the given T_EX color. (Spot colors are forced to cmyk model, so this operator is not recommended for spot colors.) Therefore the example shown above would raise a METAPOST error: `cmykcolor col;` should have been declared. By contrast, `mplibrgbtexcolor` (*string*) always returns rgb model expressions.

1.2.3 `mplibgraphictext ...`

`mplibgraphictext` is a METAPOST operator, the effect of which is similar to that of ConT_EXt's `graphictext` or our own `mpliboutlinetext` (see below 1.2.6). However the syntax is somewhat different.

```
draw mplibgraphictext "Funny"
  fakebold 2.3                % fontspec option
  drawcolor .7blue fillcolor "red!50" % color expressions
;
```

`fakebold`, `drawcolor` and `fillcolor` are optional; default values are 2, "black" and "white" respectively. When the color expressions are given in string type, they are regarded as `color`, `xcolor` or `l3color`'s expressions. All from `mplibgraphictext` to the end of sentence will compose an anonymous picture, which can be drawn or assigned to a variable. Incidentally, `withdrawcolor` and `withfillcolor` are synonyms of `drawcolor` and `fillcolor`, hopefully to be compatible with `graphictext`.

N.B. In some cases, `mplibgraphictext` will produce better results than ConT_EXt or even than our own `mpliboutlinetext`, especially when processing complicated T_EX code such as the vertical writing in Chinese or Japanese. However, because the implementation is quite different from others, there are some limitations such that you can't apply shading (gradient colors) to the text with *metafun*'s `withshademethod`.⁸ Again, in DVI mode, `unicode-math` package is needed for math formulae, as we cannot embolden type1 fonts in DVI mode.

1.2.4 `mplibglyph ... of ...`

From v2.30, we provide a new METAPOST operator `mplibglyph`, which returns a METAPOST picture containing outline paths of a glyph in opentype, truetype or type1 fonts. When a type1 font is specified, METAPOST primitive `glyph` will be called.

```
mplibglyph 50 of \fontid\font          % slot 50 of current font
mplibglyph "Q" of "TU/TeXGyrePagella(0)/m/n/10" % font csname
```

⁸But this limitation is now lifted by the introduction of `withshadingmethod`. See below 1.2.12.

```


mplibglyph "Q" of "texgyrepagella-regular.otf" % raw filename
mplibglyph "Q" of "Times.ttc(2)" % subfont number
mplibglyph "Q" of "SourceHanSansK-VF.otf[Regular]" % instance name

```

Both arguments before and after of “of” can be either a number or a string. Number arguments are regarded as a glyph slot (GID) and a font id number, respectively. String argument at the left side is regarded as a glyph name in the font or a unicode character. String argument at the right side is regarded as a \TeX font csname (without backslash) or the raw filename of a font. When it is a font filename, a number within parentheses after the filename denotes a subfont number (starting from zero) of a TTC font; a string within brackets denotes an instance name of a variable font.

1.2.5 mplibdrawglyph ...

The picture returned by `mplibglyph` will be quite similar to the result of `glyph` primitive in its structure. So, `METAPOST`’s `draw` command will fill the inner path of the picture with the background color. In contrast, `mplibdrawglyph` $\langle picture \rangle$ command fills the paths according to the nonzero winding number rule. As a result, for instance, the area surrounded by inner path of “O” will remain transparent.

 To apply the nonzero winding number rule to a picture containing paths, `luamplib` appends `withpostscript "collect"` to the paths except the last one in the picture. If you want the even-odd rule instead, you can, with *plain* format as well, additionally declare `withpostscript "evenodd"` to the last path in the picture.

1.2.6 mpliboutlinetext (...)

From v2.31, a new `METAPOST` operator `mpliboutlinetext` is available, which mimicks *metafun*’s `outlinetext`. So the syntax is the same: see the *metafun* manual § 8.7 (texdoc metafun). A simple example:

```

draw mpliboutlinetext.b ("$\sqrt{2+\alpha}$")
  (withcolor \mpcolor{red!50})
  (withpen pencircle scaled .2 withcolor red)
  scaled 2 ;

```

After the process, `mpliboutlinepic[]` and `mpliboutlinenum` will be preserved as global variables; `mpliboutlinepic[1] ... mpliboutlinepic[mpliboutlinenum]` will be an array of images each of which containing a glyph or a rule.

N.B. As Unicode grapheme cluster is not considered in the array, a unit that must be a single cluster might be separated apart.

1.2.7 \mppattern{...} ... \endmppattern, ... withmppattern ...

\TeX macros `\mppattern{<name>} ... \endmppattern` define a tiling pattern associated with the $\langle name \rangle$. `METAPOST` command `withmppattern`, the syntax being $\langle path \rangle | \langle textual picture \rangle$ `withmppattern` $\langle string \rangle$, will fill the given path or text with the tiling pattern of the $\langle name \rangle$

by replicating it horizontally and vertically.⁹ The *textual picture* here means any text typeset by T_EX, mostly the result of the `btex` command (though technically this is not a true textual picture) or the `infont` operator.

An example:

```
\mppattern{mypatt}           % or \begin{mppattern}{mypatt}
[                             % options: see below
  xstep = 10,
  ystep = 12,
  matrix = {0, 1, -1, 0},     % or "0 1 -1 0" or "rotated 90"
]
\mpfig                       % or any other TeX code,
  draw (origin--(1,1))
    scaled 10
    withcolor 1/3[blue,white]
  ;
  draw (up--right)
    scaled 10
    withcolor 1/3[red,white]
  ;
\endmpfig
\endmppattern                % or \end{mppattern}

\mpfig
  draw fullcircle scaled 90
    withpostscript "collect"
  ;
  filldraw fullcircle scaled 200
    withmppattern "mypatt"
    withpen pencircle scaled 1
    withcolor \mpcolor{red!50!blue!50}
    withpostscript "evenodd"
  ;
\endmpfig
```

The available options are listed in Table 1.

For the sake of convenience, the width and height values of tiling patterns will be written down into the log file. (depth is always zero.) Users can refer to them for option setting.

As for `matrix` option, METAPOST code such as `"rotated 30 slanted .2"` is allowed as well as string or table of four numbers. You can also set `xshift` and `yshift` values by using ‘shifted’ operator. But when `xshift` or `yshift` option is explicitly given, they have precedence over the effect of ‘shifted’ operator.

When you use special effects such as transparency in a pattern, `resources` option is needed: for instance, `resources="/ExtGState 1 0 R"`. However, as `luamplib` automatically includes the resources of the current page, this option is not needed in most cases.

⁹`withpattern` is an operator virtually the same as `withmppattern`, but the former forces a METAPOST picture. So users cannot use the drawing commands such as `fill` or `filldraw` with `withpattern` operator.

Table 1: options for \mppattern

Key	Value Type	Explanation
xstep	<i>number</i>	horizontal spacing between pattern cells
ystep	<i>number</i>	vertical spacing between pattern cells
xshift	<i>number</i>	horizontal shifting of pattern cells
yshift	<i>number</i>	vertical shifting of pattern cells
bbox	<i>table or string</i>	llx, lly, urx, ury values*
matrix	<i>table or string</i>	xx, yx, xy, yy values* or MP transform code
resources	<i>string</i>	PDF resources if needed
colored or coloured	<i>boolean</i>	false for uncolored pattern. default: true

* in string type, numbers are separated by spaces

Option colored=false (or coloured=false) will generate an uncolored pattern which shall have no color at all. Uncolored pattern will be painted later by the color of a METAPOST object. An example:

```

\begin{mppattern}{pattnocolor}
[
  colored = false,
  matrix = "slanted .3 rotated 30",
]
\tiny\TeX
\end{mppattern}

\begin{mplibcode}
beginfig(1)
picture tex;
tex = mpliboutlinetext.p ("bfseries \TeX");
for i=1 upto mpliboutlinenum:
  j:=0;
  for item within mpliboutlinepic[i]:
    j:=j+1;
    filldraw pathpart item scaled 10
    if j < length mpliboutlinepic[i]:
      withpostscript "collect"
    else:
      withmppattern "pattnocolor"
      withpen pencircle scaled 1/2
      withcolor (i/4)[red,blue]          % paints the pattern
    fi;
  endfor
endfor
endfig;
\end{mplibcode}

```

A much simpler and efficient way to obtain a similar result (without colorful characters in this

example) is to give a *textual picture* as the operand of `withmppattern`:

```
\begin{mplibcode}
  beginfig(2)
  draw mplibgraphicstext "\bfseries\TeX"
    fakebold 1
    fillcolor 1/3[red,blue]          % paints the pattern
    drawcolor 2/3[red,blue]
    scaled 10
    withmppattern "pattnocolor" ;
  endfig;
\end{mplibcode}
```

1.2.8 ... `withfademethod` ...

This is a METAPOST operator which makes the color of an object gradiently transparent. The syntax is $\langle path \rangle | \langle picture \rangle$ `withfademethod` $\langle string \rangle$, the latter being either "linear" or "circular". Though it is similar to the `withshademethod` from *metafun*, the differences are: (1) the operand of `withfademethod` can be a picture as well as a path; (2) you cannot make gradient colors, but can only make gradient opacity.

Related macros to control optional values are:

`withfadeopacity` (*number, number*) sets the starting opacity and the ending opacity, default value being (1,0). '1' denotes full color; '0' full transparency.

`withfadevector` (*pair, pair*) sets the starting and ending points. Default value in the linear mode is (llcorner p, lrcorner p), where p is the operand, meaning that fading starts from the left edge and ends at the right edge. Default value in the circular mode is (center p, center p), which means centers of both starting and ending circles are the center of the bounding box.

`withfadecenter` is a synonym of `withfadevector`.

`withfaderadius` (*number, number*) sets the radii of starting and ending circles. This is no-op in the linear mode. Default value is (0, abs(center p - urcorner p)), meaning that fading starts from the center and ends at the four corners of the bounding box.

`withfadebbox` (*pair, pair*) sets the bounding box of the fading area, default value being (llcorner p, urcorner p). Though this option is not needed in most cases, there could be cases when users want to explicitly control the bounding box. Particularly, see the description [below](#) on the analogous macro `withgroupbbox`.

An example:

```
\mpfig
  picture mill;
  mill = btex \includegraphics[width=100bp]{mill} etex;
  draw mill
```

```

withfademethod "circular"
withfadecenter (center mill, center mill)
withfaderadius (20, 50)
withfadeopacity (1, 0)
;
\endmpfig

```

1.2.9 ... asgroup ...

As said [before](#), transparency group is available with *plain* as well as *metafun* format. The syntax is exactly the same: $\langle picture \rangle | \langle path \rangle$ asgroup "" | "isolated" | "knockout" | "isolated,knockout", which will return a METAPOST picture. It is called *Transparency Group* because the objects contained in the group are composited to produce a single object, so that outer transparency effect, if any, will be applied to the group as a whole, not to the individual objects cumulatively.

The additional feature provided by *luamplib* is that you can reuse the group as many times as you want in the \TeX code or in other METAPOST code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide \TeX and METAPOST macros as follows:

`withgroupname $\langle string \rangle$` associates a transparency group with the given name. When this is not appended to the sentence with asgroup operator, the default group name 'lastmplibgroup' will be used.

`\usemplibgroup{ $\langle name \rangle$ }` is a \TeX command to reuse a transparency group of the name once used. Note that the position of the group will be origin-based: in other words, lower-left corner of the group will be shifted to the origin.

`usemplibgroup $\langle string \rangle$` is a METAPOST command which will add a transparency group of the name to the currentpicture. Contrary to the \TeX command just mentioned, the position of the group is the same as the original transparency group.

`withgroupbbox ($pair, pair$)` sets the bounding box of the transparency group, default value being (llcorner p, urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence 'withgroupbbox (bot lft llcorner p, top rt urcorner p)', supposing that the pen was selected by the pickup command.

An example showing the difference between the \TeX and METAPOST commands:

```

\mpfig
draw image(
  fill fullcircle scaled 100 shifted 25right withcolor blue;
  fill fullcircle scaled 100 withcolor red ;
) asgroup ""
  withgroupname "mygroup";
draw (left--right) scaled 10;
draw (up--down) scaled 10;
\endmpfig

```



```

\noindent
\clap{\vrule width 20pt height .25pt depth .25pt}%
\clap{\vrule width .5pt height 10pt depth 10pt}%
\usemplibgroup{mygroup}

\mpfig
  usemplibgroup "mygroup" rotated 15
  withtransparency (1, 0.5) ;
  draw (left--right) scaled 10;
  draw (up--down) scaled 10;
\endmpfig

```

Also note that normally the reused transparency groups are not affected by outer color commands. However, if you have made the original transparency group using `withoutcolor` command, colors will have effects on the uncolored objects in the group.

1.2.10 `\mplibgroup{...}` ... `\endmplibgroup`

These \TeX macros are described here in this subsection, as they are deeply related to the `asgroup` operator. Users can define a transparency group or a normal *form XObject* with these macros from \TeX side. The syntax is similar to the `\mppattern` command (see above 1.2.7). An example:

```

\mplibgroup{mygrx}                % or \begin{mplibgroup}{mygrx}
[                                % options: see below
  asgroup="",
]
\mpfig                            % or any other TeX code
  pickup pencircle scaled 10;
  draw (left--right) scaled 30 rotated 45 ;
  draw (left--right) scaled 30 rotated -45 ;
\endmpfig
\endmplibgroup                    % or \end{mplibgroup}

\usemplibgroup{mygrx}

\mpfig
  usemplibgroup "mygrx" scaled 1.5
  withtransparency (1, 0.5) ;
\endmpfig

```

Available options, much fewer than those for `\mppattern`, are listed in Table 2. Again, the width/height/depth values of the `mplibgroup` will be written down into the log file.

When `asgroup` option, including empty string, is not given, a normal *form XObject* will be generated rather than a transparency group. Thus the individual objects, not the *XObject* as a whole, will be affected by outer transparency command.

As shown, you can reuse the `mplibgroup` using the \TeX command `\usemplibgroup` or the `METAPOST` command `usemplibgroup`. The behavior of these commands is the same as that described above, excepting that the `mplibgroup` made by \TeX code (not by `METAPOST` code) respects original height and depth.

Table 2: options for \mplibgroup

Key	Value Type	Explanation
asgroup	<i>string</i>	"" , "isolated" , "knockout" , or "isolated, knockout"
bbox	<i>table</i> or <i>string</i>	llx, lly, urx, ury values*
matrix	<i>table</i> or <i>string</i>	xx, yx, xy, yy values* or MP transform code
resources	<i>string</i>	PDF resources if needed

* in string type, numbers are separated by spaces

1.2.11 ... withtransparency ...

withtransparency(*number* | *string*, *number*) is provided for *plain* format as well. The first argument accepts a number or a name of alternative transparency methods (see texdoc metafun § 8.2 Figure 8.1). The second argument accepts a number denoting opacity.

```
fill fullcircle scaled 10
  withcolor red
  withtransparency (1, 0.5)      % or ("normal", 0.5)
;
```

1.2.12 ... withshadingmethod ...

The syntax is exactly the same as *metafun*'s new shading method (texdoc metafun § 8.3.3), except that the 'shade' contained in each and every macro name has changed to 'shading' in *luamplib*: for instance, while withshademethod is a macro name which only works with *metafun* format, the equivalent provided by *luamplib*, withshadingmethod, works with *plain* as well. Other differences to the *metafun*'s and some cautions are:

- *textual pictures* (pictures made by btex ... etex, texttext, TEX, maketext, mplibgraphictext, infont, etc) as well as paths can have shading effect.

```
draw btex \bfseries\TeX etex rotated 30 scaled 10
  withshadingmethod "linear"
  withshadingvector (0,1)
  withshadingstep (
    withshadingfraction 1/2
    withshadingcolors (red,green)
  )
  withshadingstep (
    withshadingfraction 1
    withshadingcolors (green,blue)
  )
;
```

- When you give shading effect to a picture made by 'infont' operator, the result of withshadingvector will be the same as that of withshadingdirection, as *luamplib* considers only the bounding box of the picture in this case.

Macros provided by `luamplib` are:

`<path> | <textual picture>` `withshadingmethod <string>` where `<string>` shall be either "linear" or "circular". This is the only 'must' item to get shading effect; all the macros below are optional.

`withshadingvector <pair>` Starting and ending points (as time value) on the path.

`withshadingdirection <pair>` Starting and ending points (as time value) on the bounding box.
Default value: `(0,2)`

`withshadingorigin <pair>` The center of starting and ending circles. Default value: center `p`

`withshadingradius <pair>` Radii of starting and ending circles. This is no-op in linear mode.
Default value: `(0, abs(center p - urcorner p))`

`withshadingfactor <number>` Multiplier of the radii. This is no-op in linear mode. Default value: 1.2

`withshadingcenter <pair>` Values for shifting starting center. For instance, `(0,0)` means that the center of starting circle is center `p`; `(1,1)` means `urcorner p`; `(-1,-1)` means `llcorner p`.

`withshadingtransform <string>` where `<string>` shall be "yes" (respect transform) or "no" (ignore transform). Default value: "no" for pictures made by `infont` operator; "yes" for all other cases.

`withshadingdomain <pair>` Limiting values of parametric variable that varies on the axis of color gradient. Default value: `(0,1)`

`withshadingstep (...)` for combined shading of more than two colors.

`withshadingfraction <number>` Fractional number of each shading step. Only meaningful with `withshadingstep`.

`withshadingcolors (color expr, color expr)` Starting and ending colors. Default value is `(white, black)`

1.2.13 `mpliblength ...`, `mplibuclength ...`

`mpliblength <string>` returns the number of unicode characters in the string. This is a unicode-aware version equivalent to the `METAPOST` primitive `length`, but accepts only a string-type argument. For instance, `mpliblength "abçdéf"` returns 6, not 8.

On the other hand, `mplibuclength <string>` returns the number of unicode grapheme clusters in the string. For instance, `mplibuclength "Äpfel"`, where `Ä` is encoded using two codepoints (`U+0041` and `U+0308`), returns 5, not 6 or 7. This operator requires `lua-uni-algos` package.

1.2.14 `mplibsubstring ... of ...`, `mplibucsubstring ... of ...`

`mplibsubstring` $\langle pair \rangle$ of $\langle string \rangle$ is a unicode-aware version equivalent to the METAPOST's `substring ... of ...` primitive. The syntax is the same as the latter, but the string is indexed by unicode characters. For instance, `mplibsubstring (2,5) of "abçdéf"` returns "çdé", and `mplibsubstring (5,2) of "abçdéf"` returns "édç".

On the other hand, `mplibucsubstring` $\langle pair \rangle$ of $\langle string \rangle$ returns the part of the string indexed by unicode grapheme clusters. For instance, `mplibucsubstring (0,1) of "Äpfel"`, where Ä is encoded using two codepoints (U+0041 and U+0308), returns "Ä", not "A". This operator requires `lua-uni-algos` package.

1.3 Lua

1.3.1 `runscript ...`

Using the primitive `runscript` $\langle string \rangle$, you can run a Lua code chunk from METAPOST side and get some METAPOST code returned by Lua if you want. As the functionality is provided by the `mplib` library itself, `luamplib` does not have much to say about it.

One thing is worth mentioning, however: if you return a Lua *table* to the METAPOST process, it is automatically converted to a relevant METAPOST value type such as `pair`, `color`, `cmykcolor` or `transform`. So users can save some extra toil of converting a table to a string, though it's not a big deal. For instance, `runscript "return {1,0,0}"` will give you the METAPOST color expression `(1,0,0)` automatically.

1.3.2 Lua table `luamplib.instances`

Users can access the Lua table containing `mplib` instances, `luamplib.instances`, through which METAPOST variables are also easily accessible from Lua side, as documented in LuaTeX manual § 11.2.8.4 (texdoc `luatex`). The following will print `false`, `3.0`, `MetaPost` and the knots and the cyclicity of the path `unitsquare`, consecutively.

```
\begin{mplibcode}[instance1]
  boolean b; b = 1 > 2;
  numeric n; n = 3;
  string s; s = "MetaPost";
  path p; p = unitsquare;
\end{mplibcode}

\directlua{
  local instance1 = luamplib.instances.instance1
  print( instance1:get_boolean "b" )
  print( instance1:get_number "n" )
  print( instance1:get_string "s" )
  local t = instance1:get_path "p"
  for k,v in pairs(t) do
    print(k, type(v)=='table' and table.concat(v, ' ') or v)
  end
}
```

Table 3: elements in luamplib table (partial)

Key	Type	Related T _E X macro
codeinherit	<i>boolean</i>	<code>\mplibcodeinherit</code>
everyendmplib	<i>table</i>	<code>\everyendmplib</code>
everymplib	<i>table</i>	<code>\everymplib</code>
getcachedir	<i>function</i> ($\langle string \rangle$)	<code>\mplibcachedir</code>
globaltexttext	<i>boolean</i>	<code>\mplibglobaltexttext</code>
legacyverbatimtex	<i>boolean</i>	<code>\mpliblegacybehavior</code>
noneedtoreplace	<i>table</i>	<code>\mplibmakenocache</code>
numbersystem	<i>string</i>	<code>\mplibnumbersystem</code>
setformat	<i>function</i> ($\langle string \rangle$)	<code>\mplibsetformat</code>
showlog	<i>boolean</i>	<code>\mplibshowlog</code>
texttextlabel	<i>boolean</i>	<code>\mplibtexttextlabel</code>
verbatiminput	<i>boolean</i>	<code>\mplibverbatim</code>

1.3.3 Lua function `luamplib.process_mplibcode`

Users can execute a METAPOST code chunk from Lua side by using this function:

```
luamplib.process_mplibcode (<string> metapost code, <string> instance name)
```

The second argument cannot be absent, but can be an empty string (`""`) which means that it has no instance name.

Some other elements in the `luamplib` namespace, listed in Table 3, can have effects on the process of `process_mplibcode`.

2 Implementation

2.1 Lua module

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.37.7",
5   date      = "2025/12/05",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8
```

Use the `luamplib` namespace, since `mplib` is for the METAPOST library itself. ConT_EXt uses `metapost`.

```

9 luamplib      = luamplib or { }
10 local luamplib = luamplib
11
12 local format, abs = string.format, math.abs
13
```

Use our own function for warn/info/err.

```

14 local function termorlog (target, text, kind)
15   if text then
16     local mod, write, append = "luamplib", texio.write_nl, texio.write
17     kind = kind
18     or target == "term" and "Warning (more info in the log)"
19     or target == "log" and "Info"
20     or target == "term and log" and "Warning"
21     or "Error"
22     target = kind == "Error" and "term and log" or target
23     local t = text:explode"\n+"
24     write(target, format("Module %s %s:", mod, kind))
25     if #t == 1 then
26       append(target, format(" %s", t[1]))
27     else
28       for _,line in ipairs(t) do
29         write(target, line)
30       end
31       write(target, format("(%s) ", mod))
32     end
33     append(target, format(" on input line %s", tex.inputlineno))
34     write(target, "")
35     if kind == "Error" then error() end
36   end
37 end
38 local function warn (...) -- beware '%' symbol
39   termorlog("term and log", select("#",...) > 1 and format(...) or ...)
40 end
41 local function info (...)
42   termorlog("log", select("#",...) > 1 and format(...) or ...)
43 end
44 local function err (...)
45   termorlog("error", select("#",...) > 1 and format(...) or ...)
46 end
47
48 luamplib.showlog = luamplib.showlog or false
49

```

This module is a stripped down version of libraries that are used by ConT_EXt. Provide a few “shortcuts” expected by the code.

```

50 local tableconcat = table.concat
51 local tableinsert = table.insert
52 local tableunpack = table.unpack
53 local texsprint   = tex.sprint
54 local texgettoks  = tex.gettoks
55 local texgetbox   = tex.getbox
56 local texruntoks  = tex.runtoks
57 if not texruntoks then
58   err("Your LuaTeX version is too old. Please upgrade it to the latest")

```

```

59 end
60 local is_defined = token.is_defined
61 local get_macro = token.get_macro
62 local mplib = require ('mplib')
63 local kpse = require ('kpse')
64 local lfs = require ('lfs')
65 local lfsattributes = lfs.attributes
66 local lfsisdir = lfs.isdir
67 local lfsmkdir = lfs.mkdir
68 local lfstouch = lfs.touch
69 local iopen = io.open
70

```

Some helper functions, prepared for the case when l-file etc is not loaded.

```

71 local file = file or { }
72 local replacesuffix = file.replacesuffix or function(filename, suffix)
73   return (filename:gsub("%.[%a%d]+$", "")) .. "." .. suffix
74 end
75 local is_writable = file.is_writable or function(name)
76   if lfsisdir(name) then
77     name = name .. "/_luam_plib_temp_file_"
78     local fh = iopen(name, "w")
79     if fh then
80       fh:close(); os.remove(name)
81       return true
82     end
83   end
84 end
85 local mk_full_path = lfs.mkdirp or lfs.mkdirs or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\\"/]+)") do
88     full = full .. sub
89     lfsmkdir(full)
90   end
91 end
92

```

btex ... etex in input .mp files will be replaced in finder. Because of the limitation of mplib regarding make_text, we might have to make cache files modified from input files.

```

93 local luamplibtime = lfsattributes(kpse.find_file"luamplib.lua", "modification")
94 local currenttime = os.time()
95 local outputdir, cachedir
96 if lfstouch then
97   for i,v in ipairs{'TEXMFVAR', 'TEXMF_OUTPUT_DIRECTORY', '.', 'TEXMFOUTPUT'} do
98     local var = i == 3 and v or kpse.var_value(v)
99     if var and var ~= "" then
100       for _,vv in next, var:explode(os.type == "unix" and ":" or ";") do
101         local dir = format("%s/%s", vv, "luamplib_cache")
102         if not lfsisdir(dir) then
103           mk_full_path(dir)

```

```

104     end
105     if is_writable(dir) then
106         outputdir = dir
107         break
108     end
109 end
110 if outputdir then break end
111 end
112 end
113 end
114 outputdir = outputdir or '.'
115 function luamplib.getcachedir(dir)
116     dir = dir:gsub("##", "#")
117     dir = dir:gsub("^~",
118         os.type == "windows" and os.getenv("UserProfile") or os.getenv("HOME"))
119     if lfstouch and dir then
120         if lfsisdir(dir) then
121             if is_writable(dir) then
122                 cachedir = dir
123             else
124                 warn("Directory '%s' is not writable!", dir)
125             end
126         else
127             warn("Directory '%s' does not exist!", dir)
128         end
129     end
130 end

```

Some basic METAPOST files not necessary to make cache files.

```

131 local noneedtoreplace = {
132     ["boxes.mp"] = true, -- ["format.mp"] = true,
133     ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,
134     ["mpost.mp"] = true, ["plain.mp"] = true, ["rboxes.mp"] = true,
135     ["sarith.mp"] = true, ["string.mp"] = true, -- ["TEX.mp"] = true,
136     ["metafun.mp"] = true, ["metafun.mpiv"] = true, ["mp-abck.mpiv"] = true,
137     ["mp-apos.mpiv"] = true, ["mp-asnc.mpiv"] = true, ["mp-bare.mpiv"] = true,
138     ["mp-base.mpiv"] = true, ["mp-blob.mpiv"] = true, ["mp-butt.mpiv"] = true,
139     ["mp-char.mpiv"] = true, ["mp-chem.mpiv"] = true, ["mp-core.mpiv"] = true,
140     ["mp-crop.mpiv"] = true, ["mp-figs.mpiv"] = true, ["mp-form.mpiv"] = true,
141     ["mp-func.mpiv"] = true, ["mp-grap.mpiv"] = true, ["mp-grid.mpiv"] = true,
142     ["mp-grph.mpiv"] = true, ["mp-idea.mpiv"] = true, ["mp-luas.mpiv"] = true,
143     ["mp-mlib.mpiv"] = true, ["mp-node.mpiv"] = true, ["mp-page.mpiv"] = true,
144     ["mp-shap.mpiv"] = true, ["mp-step.mpiv"] = true, ["mp-text.mpiv"] = true,
145     ["mp-tool.mpiv"] = true, ["mp-cont.mpiv"] = true,
146 }
147 luamplib.noneedtoreplace = noneedtoreplace

```

format.mp is much complicated, so specially treated.

```

148 local function replaceformatmp(file,newfile,ofmodify)
149     local fh = ioopen(file,"r")

```



```

150 if not fh then return file end
151 local data = fh:read("*all"); fh:close()
152 fh = ioopen(newfile,"w")
153 if not fh then return file end
154 fh:write(
155     "let normalinfont = infont;\n",
156     "primarydef str infont name = rawtexttext(str) enddef;\n",
157     data,
158     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
159     "vardef Fexp_(expr x) = rawtexttext("\$^{\"&decimal x&\"}$\") enddef;\n",
160     "let infont = normalinfont;\n"
161 ); fh:close()
162 lfstouch(newfile,currenttime,ofmodify)
163 return newfile
164 end

```

Replace `btex ... etex` and `verbatimtex ... etex` in input files, if needed.

```

165 local name_b = "%f[%a_]"
166 local name_e = "%f[^%a_]"
167 local btex_etex = name_b.."btex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
168 local verbatimtex_etex = name_b.."verbatimtex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
169 local function replaceinputmpfile (name,file)
170     local ofmodify = lfsattributes(file,"modification")
171     if not ofmodify then return file end
172     local newfile = name:gsub("%W","_")
173     newfile = format("%s/luamplib_input_%s", cachedir or outputdir, newfile)
174     if newfile and luamplibtime then
175         local nf = lfsattributes(newfile)
176         if nf and nf.mode == "file" and
177             ofmodify == nf.modification and luamplibtime < nf.access then
178             return nf.size == 0 and file or newfile
179         end
180     end
181     if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
182     local fh = ioopen(file,"r")
183     if not fh then return file end
184     local data = fh:read("*all"); fh:close()

```

“`etex`” must be preceded by a space and followed by a space or semicolon as specified in `LuaTeX` manual, which is not the case of standalone `METAPOST` though.

```

185 local count,cnt = 0,0
186 data, cnt = data:gsub(btex_etex, "btex %1 etex ") -- space
187 count = count + cnt
188 data, cnt = data:gsub(verbatimtex_etex, "verbatimtex %1 etex;") -- semicolon
189 count = count + cnt
190 if count == 0 then
191     needtoreplace[name] = true
192     fh = ioopen(newfile,"w");
193     if fh then
194         fh:close()

```

```

195     lfstouch(newfile,currenttime,ofmodify)
196   end
197   return file
198 end
199 fh = ioopen(newfile,"w")
200 if not fh then return file end
201 fh:write(data); fh:close()
202 lfstouch(newfile,currenttime,ofmodify)
203 return newfile
204 end
205

```

As the finder function for mplib, use the kpse library and make it behave like as if METAPOST was used. And replace .mp files with cache files if needed. See also #74, #97.

```

206 local mpkpse
207 do
208   local exe = 0
209   while arg[exe-1] do
210     exe = exe-1
211   end
212   mpkpse = kpse.new(arg[exe], "mpost")
213 end
214 local special_ftype = {
215   pfb = "type1 fonts",
216   enc = "enc files",
217 }
218 function luamplib.finder (name, mode, ftype)
219   if mode == "w" then
220     if name and name ~= "mpout.log" then
221       kpse.record_output_file(name) -- recorder
222     end
223     return name
224   else
225     ftype = special_ftype[ftype] or ftype
226     local file = mpkpse.find_file(name,ftype)
227     if file then
228       if lfstouch and ftype == "mp" and not noneedtoreplace[name] then
229         file = replaceinputmpfile(name,file)
230       end
231     else
232       file = mpkpse.find_file(name, name:match("%a+$"))
233     end
234     if file then
235       kpse.record_input_file(file) -- recorder
236     end
237     return file
238   end
239 end
240

```

Create and load `mplib` instances. We do not support ancient version of `mplib` any more. (Don't know which version of `mplib` started to support `make_text` and `run_script`; let the users find it.)

```

241 local preamble = [[
242   boolean mplib ; mplib := true ;
243   let dump = endinput ;
244   let normalfontsize = fontsize;
245   input %s ;
246 ]]

```

plain or *metafun*, though we cannot support *metafun* format fully.

```

247 local currentformat = "plain"
248 function luamplib.setformat (name)
249   currentformat = name
250 end

```

v2.9 has introduced the concept of “code inherit”

```

251 luamplib.codeinherit = false
252 local mplibinstances = {}
253 luamplib.instances = mplibinstances
254 local has_instancename = false
255 local function reporterror (result, prevlog)
256   if not result then
257     err("no result object returned")
258   else
259     local t, e, l = result.term, result.error, result.log

```

`log` has more information than `term`, so `log` first (2021/08/02)

```

260   local log = l or t or "no-term"
261   log = log:gsub("%(Please type a command or say `end'%)", ""):gsub("\n+", "\n")
262   if result.status > 0 then
263     local first = log:match("(-\n! .-)\n! "
264     if first then
265       termorlog("term", first)
266       termorlog("log", log, "Warning")
267     else
268       warn(log)
269     end
270     if result.status > 1 then
271       err(e or "see above messages")
272     end
273   elseif prevlog then
274     log = prevlog..log

```

v2.6.1: now `luamplib` does not disregard `show` command, even when `luamplib.showlog` is false.

Incidentally, it does not raise error nor prints an info, even if output has no figure.

```

275   local show = log:match"\n>>? .+"
276   if show then
277     termorlog("term", show, "Info (more info in the log)")
278     info(log)
279   elseif luamplib.showlog and log:find"%g" then

```

```

280     info(log)
281   end
282 end
283 return log
284 end
285 end

```

lualibs-os.lua installs a randomseed. When this file is not loaded, we should explicitly seed a unique integer to get random randomseed for each run.

```

286 if not math.initialseed then math.randomseed(currenttime) end
287 local function luamplibload (name)
288   local mpx = mplib.new {
289     ini_version = true,
290     find_file   = luamplib.finder,

```

Make use of `make_text` and `run_script`, which will co-operate with LuaTeX's `tex.runtoks` or other Lua functions. And we provide `numbersystem` option since v2.4. See <https://github.com/lualatex/luamplib/issues/21>.

```

291   make_text   = luamplib.maketext,
292   run_script  = luamplib.runscript,
293   math_mode   = luamplib.numbersystem,
294   job_name    = tex.jobname,
295   random_seed = math.random(4095),
296   utf8_mode   = true,
297   extensions  = 1,
298 }

```

Append our own METAPOST preamble to the preamble above.

```

299 local preamble = tableconcat{
300   format(preamble, replacesuffix(name,"mp")),
301   luamplib.preambles.mplibcode,
302   luamplib.legacyverbatimtex and luamplib.preambles.legacyverbatimtex or "",
303   luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
304 }
305 local result, log
306 if not mpx then
307   result = { status = 99, error = "out of memory"}
308 else
309   result = mpx:execute(preamble)
310 end
311 log = reporterror(result)
312 return mpx, result, log
313 end

```

Here, excute each `mplibcode` data, ie `\begin{mplibcode} ... \end{mplibcode}`.

```

314 local function process (data, instancename)
315   local currfmt
316   if instancename and instancename ~= "" then
317     currfmt = instancename
318     has_instancename = true
319   else

```

```

320     currfmt = tableconcat{
321         currentformat,
322         luamplib.numbersystem or "scaled",
323         tostring(luamplib.texttextlabel),
324         tostring(luamplib.legacyverbatimimtex),
325     }
326     has_instancename = false
327 end
328 local mpx = mplibinstances[currfmt]
329 local standalone = not (has_instancename or luamplib.codeinherit)
330 if mpx and standalone then
331     mpx:finish()
332 end
333 local log = ""
334 if standalone or not mpx then
335     mpx, _, log = luamplibload(currentformat)
336     mplibinstances[currfmt] = mpx
337 end
338 local converted, result = false, {}
339 if mpx and data then
340     result = mpx:execute(data)
341     local log = reporterror(result, log)
342     if log then
343         if result.fig then
344             converted = luamplib.convert(result)
345         end
346     end
347 else
348     err"Mem file unloadable. Maybe generated with a different version of mplib?"
349 end
350 return converted, result
351 end
352

```

dvipdfmx is supported, though nobody seems to use it.

```

353 local pdfmode = tex.outputmode > 0
354

```

make_text and some run_script uses Lua_{TeX}'s tex.runtoks.

```

355 local catlatex = luatexbase.registernumber("catcodetable@latex")
356 local catat11 = luatexbase.registernumber("catcodetable@atletter")

```

tex.scantoks sometimes fail to read catcode properly, especially \#, \&, or \%. After some experiment, we dropped using it. Instead, a function containing tex.sprint seems to work nicely.

```

357 local function run_tex_code (str, cat)
358     texruntoks(function() texsprint(cat or catlatex, str) end)
359 end

```

Prepare texttext box number containers, locals and globals. localid can be any number. They are local anyway. The number will be reset at the start of a new code chunk. Global

boxes will use `\newbox` command in `tex.runtoks` process. This is the same when `codeinherit` is true. Boxes in instances with name will also be global, so that their tex boxes can be shared among instances of the same name.

```
360 local texboxes = { globalid = 0, localid = 4096 }
```

For conversion of sp to bp.

```
361 local factor = 65536*(7227/7200)
362 local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
363   xscaled %f yscaled %f shifted (0,-%f) \z
364   withprescript "mplibtexboxid=%i:%f:%f")'
365 local function process_tex_text (str, maketext)
366   if str then
367     if not maketext then str = str:gsub("\r.-$", "") end
368     local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
369                   and "\global" or ""
370     local tex_box_id
371     if global == "" then
372       tex_box_id = texboxes.localid + 1
373       texboxes.localid = tex_box_id
374     else
375       local boxid = texboxes.globalid + 1
376       texboxes.globalid = boxid
377       run_tex_code(format([[ \expandafter \newbox \csname luamplib.box.%s \endcsname ]], boxid))
378       tex_box_id = tex.getcount 'allocationnumber'
379     end
380     if str:find"^[taggingoff%]" then
381       str = str:gsub("^[taggingoff%]*s*", "")
382       run_tex_code(format("\luamplibnotagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
383                           tex_box_id, global, tex_box_id, str))
384     else
385       run_tex_code(format("\luamplibtagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
386                           tex_box_id, global, tex_box_id, str))
387     end
388     local box = texgetbox(tex_box_id)
389     local wd = box.width / factor
390     local ht = box.height / factor
391     local dp = box.depth / factor
392     return texttext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
393   end
394   return ""
395 end
396
```

Make `color` or `xcolor`'s color expressions usable, with `\mpcolor` or `mplibcolor`. These commands should be used with graphical objects. Attempt to support `l3color` as well.

```
397 local mplibcolorfmt = {
398   xcolor = tableconcat{
399     [[ \begingroup \let \XC@mcolor \relax ],
400     [[ \def \set@color {\global \mplibtmptoks \expandafter {\current@color}} ]],
401     [[ \color %s \endgroup ]],

```

```

402 },
403 l3color = tableconcat{
404   [[\begingroup\def\__color_select:N#1{\expandafter\__color_select:nn#1}]],
405   [[\def\__color_backend_select:nn#1#2{\global\mplibtmptoks{#1 #2}}]],
406   [[\def\__kernel_backend_literal:e#1{\global\mplibtmptoks\expandafter{\expanded{#1}}}],
407   [[\color_select:n%s\endgroup]],
408 },
409 }
410 local colfmt = is_defined'color_select:n' and "l3color" or "xcolor"
411 if colfmt == "l3color" then
412   run_tex_code{
413     "\newcatcodetable\luamplibcctabexplat",
414     "\begingroup",
415     "\catcode\@=11 ",
416     "\catcode\_ =11 ",
417     "\catcode\:=11 ",
418     "\savecatcodetable\luamplibcctabexplat",
419     "\endgroup",
420   }
421 end
422 local ccexplat = luatexbase.registernumber"luamplibcctabexplat"
423 local function process_color (str)
424   if str then
425     if not str:find("%b{") then
426       str = format("{%s}",str)
427     end
428     local myfmt = mplibcolorfmt[colfmt]
429     if colfmt == "l3color" and is_defined"color" then
430       if str:find("%b[") then
431         myfmt = mplibcolorfmt.xcolor
432       else
433         for _,v in ipairs(str:match"{{(.+)}}:explode"!") do
434           if not v:find("^%s*d+%s*$") then
435             local pp = get_macro(format("l__color_named_%s_prop",v))
436             if not pp or pp == "" then
437               myfmt = mplibcolorfmt.xcolor
438             break
439           end
440         end
441       end
442     end
443     run_tex_code(myfmt:format(str), ccexplat or catat11)
444     local t = texgettoks"mplibtmptoks"
445     if not pdfmode and not t:find"^pdf" then
446       t = t:gsub("%a+ (.+)", "pdf:bc [%1]")
447     end
448     return format('1 withprescript "mpliboverridecolor=%s"', t)
449   end
450 end

```

```

451 return ""
452 end
453
    for \mpdim or mplibdimen
454 local function process_dimen (str)
455   if str then
456     str = str:gsub("{(.+)}", "%1")
457     run_tex_code(format([[\\mplibtmp toks\\expandafter{\\the\\dimexpr %s\\relax}]], str))
458     return format("begin group %s end group", texgettoks"mplibtmp toks")
459   end
460   return ""
461 end
462

```

Newly introduced method of processing verbatimtex ... etex. This function is used when `\\mpliblegacybehavior{false}` is declared.

```

463 local function process_verbatimtex_text (str)
464   if str then
465     run_tex_code(str)
466   end
467   return ""
468 end
469

```

For legacy verbatimtex process. verbatimtex ... etex before `beginfig()` is not ignored, but the \TeX code is inserted just before the `mplib` box. And \TeX code inside `beginfig()` ... `endfig` is inserted after the `mplib` box.

```

470 local tex_code_pre_mplib = {}
471 luamplib.figid = 1
472 luamplib.in_the_fig = false
473 local function process_verbatimtex_prefig (str)
474   if str then
475     tex_code_pre_mplib[luamplib.figid] = str
476   end
477   return ""
478 end
479 local function process_verbatimtex_infig (str)
480   if str then
481     return format('special "postmplibverbtex=%s";', str)
482   end
483   return ""
484 end
485
486 local runscript_funcs = {
487   luamplibtext    = process_tex_text,
488   luamplibcolor   = process_color,
489   luamplibdimen   = process_dimen,
490   luamplibprefig  = process_verbatimtex_prefig,
491   luamplibinfig   = process_verbatimtex_infig,
492   luamplibverbtex = process_verbatimtex_text,

```



```

493 }
494

```

For *metafun* format. see issue #79.

```

495 mp = mp or {}
496 local mp = mp
497 mp.mf_path_reset = mp.mf_path_reset or function() end
498 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
499 mp.report = mp.report or info

```

metafun 2021-03-09 changes crashes luamplib.

```

500 catcodes = catcodes or {}
501 local catcodes = catcodes
502 catcodes.numbers = catcodes.numbers or {}
503 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
504 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
505 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
506 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
507 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
508 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
509 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
510

```

A function from ConT_EXt general.

```

511 local function mpprint(buffer,...)
512   for i=1,select("#",...) do
513     local value = select(i,...)
514     if value ~= nil then
515       local t = type(value)
516       if t == "number" then
517         buffer[#buffer+1] = format("%.16f",value)
518       elseif t == "string" then
519         buffer[#buffer+1] = value
520       elseif t == "table" then
521         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
522       else -- boolean or whatever
523         buffer[#buffer+1] = tostring(value)
524       end
525     end
526   end
527 end
528 function luamplib.runscript (code)
529   local id, str = code:match("(.-){(.*)}")
530   if id and str then
531     local f = runscript_funcs[id]
532     if f then
533       local t = f(str)
534       if t then return t end
535     end
536   end
537   local f = loadstring(code)

```

```

538 if type(f) == "function" then
539   local buffer = {}
540   function mp.print(...)
541     mpprint(buffer,...)
542   end
543   local res = {f()}
544   buffer = tableconcat(buffer)
545   if buffer and buffer ~= "" then
546     return buffer
547   end
548   buffer = {}
549   mpprint(buffer, tableunpack(res))
550   return tableconcat(buffer)
551 end
552 return ""
553 end
554
    make_text must be one liner, so comment sign is not allowed.
555 local function protecttexcontents (str)
556   return str:gsub("\\%", "\\0PerCent\0")
557         :gsub("%%.-\n", "")
558         :gsub("%%.-$", "")
559         :gsub("%zPerCent%z", "\\%")
560         :gsub("\r.-$", "")
561         :gsub("%s+", " ")
562 end
563 luamplib.legacyverbatimimtex = true
564 function luamplib.maketext (str, what)
565   if str and str ~= "" then
566     str = protecttexcontents(str)
567     if what == 1 then
568       if not str:find("\\documentclass"..name_e) and
569         not str:find("\\begin%s*{document}") and
570         not str:find("\\documentstyle"..name_e) and
571         not str:find("\\usepackage"..name_e) then
572         if luamplib.legacyverbatimimtex then
573           if luamplib.in_the_fig then
574             return process_verbatimimtex_infig(str)
575           else
576             return process_verbatimimtex_prefig(str)
577           end
578         else
579           return process_verbatimimtex_text(str)
580         end
581       end
582     else
583       return process_tex_text(str, true) -- bool is for 'char13'
584     end
585   end

```

```

586 return ""
587 end
588
    luamplib's METAPOST color operators
589 local function colorsplit (res)
590   local t, tt = { }, res:gsub("[%%]", "", 2):explode()
591   local be = tt[1]:find"^%d" and 1 or 2
592   for i=be, #tt do
593     if not tonumber(tt[i]) then break end
594     t[#t+1] = tt[i]
595   end
596   return t
597 end
598
599 luamplib.gettexcolor = function (str, rgb)
600   local res = process_color(str):match'"mpliboverridecolor=(.)"'
601   if res:find" cs " or res:find"@pdf.obj" then
602     if not rgb then
603       warn("%s is a spot color. Forced to CMYK", str)
604     end
605     run_tex_code({
606       "\\color_export:nnN{",
607       str,
608       "}{",
609       rgb and "space-sep-rgb" or "space-sep-cmyk",
610       "}"\\mplib_@tempa",
611     }, ccexplat)
612     return get_macro"mplib_@tempa":explode()
613   end
614   local t = colorsplit(res)
615   if #t == 3 or not rgb then return t end
616   if #t == 4 then
617     return { 1 - math.min(1, t[1]+t[4]), 1 - math.min(1, t[2]+t[4]), 1 - math.min(1, t[3]+t[4]) }
618   end
619   return { t[1], t[1], t[1] }
620 end
621
622 luamplib.shadecolor = function (str)
623   local res = process_color(str):match'"mpliboverridecolor=(.)"'
624   if res:find" cs " or res:find"@pdf.obj" then -- spot color shade: 13 only

```

An example of spot color shading:

```

\DocumentMetadata{ }
\documentclass{article}
\usepackage{luamplib}
\ExplSyntaxOn
\color_model_new:nnn { pantone3005 }
{ Separation }
{

```

```

    name = PANTONE~3005~U ,
    alternative-model = cmyk ,
    alternative-values = {1, 0.56, 0, 0}
}
\color_set:nnn{spotA}{pantone3005}{1}
\color_set:nnn{spotB}{pantone3005}{0.6}
\color_model_new:nnn { pantone1215 }
{ Separation }
{
    name = PANTONE~1215~U ,
    alternative-model = cmyk ,
    alternative-values = {0, 0.15, 0.51, 0}
}
\color_set:nnn{spotC}{pantone1215}{1}
\color_model_new:nnn { pantone2040 }
{ Separation }
{
    name = PANTONE~2040~U ,
    alternative-model = cmyk ,
    alternative-values = {0, 0.28, 0.21, 0.04}
}
\color_set:nnn{spotD}{pantone2040}{1}
\ExplSyntaxOff
\begin{document}
\begin{mplibcode}
beginfig(1)
    fill unitsquare xscaled \mpdim\textwidth yscaled 1cm
        withshadingmethod "linear"
        withshadingvector (0,1)
        withshadingstep (
            withshadingfraction .5
            withshadingcolors ("spotB","spotC")
        )
        withshadingstep (
            withshadingfraction 1
            withshadingcolors ("spotC","spotD")
        )
    ;
endfig;
\end{mplibcode}
\end{document}

```

another one: user-defined DeviceN colorspace

```

\DocumentMetadata{ }
\documentclass{article}
\usepackage{luamplib}
\ExplSyntaxOn
\color_model_new:nnn { pantone1215 }
{ Separation }

```

```

{
  name = PANTONE~1215~U ,
  alternative-model = cmyk ,
  alternative-values = {0, 0.15, 0.51, 0}
}
\color_model_new:nnn { pantone+black }
{ DeviceN }
{ names = {pantone1215,black} }
\color_set:nnn{purepantone}{pantone+black}{1,0}
\color_set:nnn{pureblack} {pantone+black}{0,1}
\ExplSyntaxOff
\begin{document}
\mpfig
fill unitsquare xscaled \mpdim{\textwidth} yscaled 30
  withshadingmethod "linear"
  withshadingcolors ("purepantone","pureblack")
;
\endmpfig
\end{document}

625 run_tex_code({
626   [[\color_export:nnN{]], str, [[]{backend}\mplib@tempa]],
627   },ccexplat)
628   local name, value = get_macro'mplib@tempa':match'{{(.-)}{(.-)}'
629   local t, obj = res:explode()
630   if pdfmode then
631     obj = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
632   else
633     obj = t[2]
634   end
635   return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
636 end
637 return colorsplit(res)
638 end
639

```

Remove trailing zeros for smaller PDF

```

640 local decimals = "%.d+"
641 local function rmzeros(str) return str:gsub("%.?0+$","") end
642

```

luamplib's mplibgraphicstext operator

```

643 local emboldenfonts = { }
644 local function getemboldenwidth (curr, fakebold)
645   local width = emboldenfonts.width
646   if not width then
647     local f
648     local function getglyph(n)
649       while n do
650         if n.head then

```

```

651         getglyph(n.head)
652     elseif n.font and n.font > 0 then
653         f = n.font; break
654     end
655     n = node.getnext(n)
656 end
657 end
658 getglyph(curr)
659 width = font.getcopy(f or font.current()).size * fakebold / factor * 10
660 emboldenfonts.width = width
661 end
662 return width
663 end
664 local function getrulewhatsit (line, wd, ht, dp)
665     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
666     local pl
667     local fmt = "%f w %f %f %f %f re %s"
668     if pdfmode then
669         pl = node.new("whatsit", "pdf_literal")
670         pl.mode = 0
671     else
672         fmt = "pdf:content " .. fmt
673         pl = node.new("whatsit", "special")
674     end
675     pl.data = fmt:format(line, 0, -dp, wd, ht+dp, "B") :gsub(decimals, rmzeros)
676     local ss = node.new"glue"
677     node.setglue(ss, 0, 65536, 65536, 2, 2)
678     pl.next = ss
679     return pl
680 end
681 local function getrulemetric (box, curr, bp)
682     local running = -1073741824
683     local wd, ht, dp = curr.width, curr.height, curr.depth
684     wd = wd == running and box.width or wd
685     ht = ht == running and box.height or ht
686     dp = dp == running and box.depth or dp
687     if bp then
688         return wd/factor, ht/factor, dp/factor
689     end
690     return wd, ht, dp
691 end

```

copying attributes of rule/glue node to improve tagging of mplibgraphicstext

```

692 local tag_update_attrs
693 if is_defined"ver@tagpdf.sty" then
694     tag_update_attrs = function (n, curr)
695         while n do
696             n.attr = curr.attr
697             if n.head then

```

```

698     tag_update_attrs(n.head, curr)
699     end
700     n = node.getnext(n)
701     end
702 end
703 else
704   tag_update_attrs = function() end
705 end
706 local function embolden (box, curr, fakebold)
707   local head = curr
708   while curr do
709     if curr.head then
710       curr.head = embolden(curr, curr.head, fakebold)
711     elseif curr.replace then
712       curr.replace = embolden(box, curr.replace, fakebold)
713     elseif curr.leader then
714       if curr.leader.head then
715         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
716       elseif curr.leader.id == node.id"rule" then
717         local glue = node.effective_glue(curr, box)
718         local line = getemboldenwidth(curr, fakebold)
719         local wd,ht,dp = getrulemetric(box, curr.leader)
720         if box.id == node.id"hlist" then
721           wd = glue
722         else
723           ht, dp = 0, glue
724         end
725         local pl = getrulewhatsit(line, wd, ht, dp)
726         local pack = box.id == node.id"hlist" and node.hpack or node.vpack
727         local list = pack(pl, glue, "exactly")
728         tag_update_attrs(list,curr)
729         head = node.insert_after(head, curr, list)
730         head, curr = node.remove(head, curr)
731       end
732     elseif curr.id == node.id"rule" and curr.subtype == 0 then
733       local line = getemboldenwidth(curr, fakebold)
734       local wd,ht,dp = getrulemetric(box, curr)
735       if box.id == node.id"vlist" then
736         ht, dp = 0, ht+dp
737       end
738       local pl = getrulewhatsit(line, wd, ht, dp)
739       local list
740       if box.id == node.id"hlist" then
741         list = node.hpack(pl, wd, "exactly")
742       else
743         list = node.vpack(pl, ht+dp, "exactly")
744       end
745       tag_update_attrs(list,curr)
746       head = node.insert_after(head, curr, list)

```

```

747     head, curr = node.remove(head, curr)
748 elseif curr.id == node.id"glyph" and curr.font > 0 then
749     local f = curr.font
750     local key = format("%s:%s",f,fakebold)
751     local i = emboldenfonts[key]
752     if not i then
753         local ft = font.getfont(f) or font.getcopy(f)
754         if pdfmode then
755             width = ft.size * fakebold / factor * 10
756             emboldenfonts.width = width
757             ft.mode, ft.width = 2, width
758             i = font.define(ft)
759         else
760             if ft.format ~= "opentype" and ft.format ~= "truetype" then
761                 goto skip_type1
762             end
763             local name = ft.name:gsub('',''):gsub(';','$','')
764             name = format('%s;embolden=%s;',name,fakebold)
765             _, i = fonts.constructors.readanddefine(name,ft.size)
766         end
767         emboldenfonts[key] = i
768     end
769     curr.font = i
770 end
771 ::skip_type1::
772 curr = node.getnext(curr)
773 end
774 return head
775 end
776 local function graphictextcolor (col, filldraw)
777     if col:find"^[%d%.:]+$" then
778         col = col:explode":"
779         for i=1,#col do
780             col[i] = format("%.3f", col[i])
781         end
782         if pdfmode then
783             local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
784             col[#col+1] = filldraw == "fill" and op or op:upper()
785             return tableconcat(col," ")
786         end
787         return format("[%s]", tableconcat(col," "))
788     end
789     col = process_color(col):match'"mpliboverridecolor=(.+)"'
790     if pdfmode then
791         local t, tt = col:explode(), { }
792         local b = filldraw == "fill" and 1 or #t/2+1
793         local e = b == 1 and #t/2 or #t
794         for i=b,e do
795             tt[#tt+1] = t[i]

```



```

796     end
797     return tableconcat(tt, " ")
798 end
799 return col:gsub("^.- ", "")
800 end
801 luamplib.graphicstext = function (text, fakebold, fc, dc)
802     local fmt = process_tex_text(text):sub(1,-2)
803     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
804     emboldenfonts.width = nil
805     local box = texgetbox(id)
806     box.head = embolden(box, box.head, fakebold)
807     local fill = graphicstextcolor(fc, "fill")
808     local draw = graphicstextcolor(dc, "draw")
809     local bc = pdfmode and "" or "pdf:bc "
810     return format('%s withprescript "mpliboverridecolor=%s%s %s"', fmt, bc, fill, draw)
811 end
812

```

luamplib's mplibglyph operator

```

813 local function mperr (str)
814     return format("hide(errmessage %q)", str)
815 end
816 local function getangle (a,b,c)
817     local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
818     if r > 180 then
819         r = r - 360
820     elseif r < -180 then
821         r = r + 360
822     end
823     return r
824 end
825 local function turning (t)
826     local r, n = 0, #t
827     for i=1,2 do
828         tableinsert(t, t[i])
829     end
830     for i=1,n do
831         r = r + getangle(t[i], t[i+1], t[i+2])
832     end
833     return r/360
834 end
835 local function glyphimage(t, fmt)
836     local q,p,r = {},{}
837     for i,v in ipairs(t) do
838         local cmd = v[#v]
839         if cmd == "m" then
840             p = {format('(%s,%s)', v[1], v[2])}
841             r = {{x=v[1], y=v[2]}}
842         else

```

```

843     local nt = t[i+1]
844     local last = not nt or nt[#nt] == "m"
845     if cmd == "l" then
846         local pt = t[i-1]
847         local seco = pt[#pt] == "m"
848         if (last or seco) and r[1].x == v[1] and r[1].y == v[2] then
849             else
850                 tableinsert(p, format('--(%s,%s)',v[1],v[2]))
851                 tableinsert(r, {x=v[1],y=v[2]})
852             end
853             if last then
854                 tableinsert(p, '--cycle')
855             end
856         elseif cmd == "c" then
857             tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
858             if last and r[1].x == v[5] and r[1].y == v[6] then
859                 tableinsert(p, '..cycle')
860             else
861                 tableinsert(p, format('..(%s,%s)',v[5],v[6]))
862                 if last then
863                     tableinsert(p, '--cycle')
864                 end
865                 tableinsert(r, {x=v[5],y=v[6]})
866             end
867         else
868             return mperr"unknown operator"
869         end
870         if last then
871             tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
872         end
873     end
874 end
875 r = { }
876 if fmt == "opentype" then
877     for _,v in ipairs(q[1]) do
878         tableinsert(r, format('addto currentpicture contour %s;',v))
879     end
880     for _,v in ipairs(q[2]) do
881         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
882     end
883 else
884     for _,v in ipairs(q[2]) do
885         tableinsert(r, format('addto currentpicture contour %s;',v))
886     end
887     for _,v in ipairs(q[1]) do
888         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
889     end
890 end
891 return format('image(%s)', tableconcat(r))

```

```

892 end
893 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
894 function luamplib.glyph (f, c)
895   local filename, subfont, instance, shapedata
896   local fid = tonumber(f) or font.id(f)
897   if fid > 0 then
898     local fontdata = font.getfont(fid) or font.getcopy(fid)
899     filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
900     instance = fontdata.specification and fontdata.specification.instance
901     filename = filename and filename:gsub("^harfloaded:", "")
902   else
903     local name
904     f = f:match"%s*(.+)%s*$"
905     name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
906     if not name then
907       name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
908     end
909     if not name then
910       name, subfont = f:match"(.+)%((%d+)%)%" -- Times.ttc(2)
911     end
912     name = name or f
913     subfont = (subfont or 0)+1
914     instance = instance and instance:lower()
915     for _,ftype in ipairs{"opentype", "truetype"} do
916       filename = kpse.find_file(name, ftype.." fonts")
917       if filename then
918         kind = ftype; break
919       end
920     end
921   end
922   if kind ~= "opentype" and kind ~= "truetype" then
923     f = fid and fid > 0 and tex.fontname(fid) or f
924     if kpse.find_file(f, "tfm") then
925       return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
926     else
927       return mperr"font not found"
928     end
929   end
930   local time = lfsattributes(filename,"modification")
931   local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
932   local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
933   local newname = format("%s/%s.lua", cachedir or outputdir, h)
934   local newtime = lfsattributes(newname,"modification") or 0
935   if time == newtime then
936     shapedata = require(newname)
937   end
938   if not shapedata then
939     shapedata = fonts and fonts.handlers.otf.readers.loadshapes(filename,subfont,instance)
940     if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end

```

```

941 table.tofile(newname, shapedata, "return")
942 lfstouch(newname, time, time)
943 end
944 local gid = tonumber(c)
945 if not gid then
946   local uni = utf8.codepoint(c)
947   for i,v in pairs(shapedata.glyphs) do
948     if c == v.name or uni == v.unicode then
949       gid = i; break
950     end
951   end
952 end
953 if not gid then return mperr"cannot get GID (glyph id)" end
954 local fac = 1000 / (shapedata.units or 1000)
955 local t = shapedata.glyphs[gid].segments
956 if not t then return "image()" end
957 for i,v in ipairs(t) do
958   if type(v) == "table" then
959     for ii,vv in ipairs(v) do
960       if type(vv) == "number" then
961         t[i][ii] = format("%.0f", vv * fac)
962       end
963     end
964   end
965 end
966 kind = shapedata.format or kind
967 return glyphimage(t, kind)
968 end
969

```

mpliboutline : based on mkiv's font-mps.lua

```

970 local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
971   unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
972 local outline_horz, outline_vert
973 function outline_vert (res, box, curr, xshift, yshift)
974   local b2u = box.dir == "LTL"
975   local dy = (b2u and -box.depth or box.height)/factor
976   local ody = dy
977   while curr do
978     if curr.id == node.id"rule" then
979       local wd, ht, dp = getrulemetric(box, curr, true)
980       local hd = ht + dp
981       if hd ~= 0 then
982         dy = dy + (b2u and dp or -ht)
983         if wd ~= 0 and curr.subtype == 0 then
984           res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
985         end
986         dy = dy + (b2u and ht or -dp)
987       end

```

```

988 elseif curr.id == node.id"glue" then
989     local vwidth = node.effective_glue(curr,box)/factor
990     if curr.leader then
991         local curr, kind = curr.leader, curr.subtype
992         if curr.id == node.id"rule" then
993             local wd = getrulemetric(box, curr, true)
994             if wd ~= 0 then
995                 local hd = vwidth
996                 local dy = dy + (b2u and 0 or -hd)
997                 if hd ~= 0 and curr.subtype == 0 then
998                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
999                 end
1000             end
1001         elseif curr.head then
1002             local hd = (curr.height + curr.depth)/factor
1003             if hd <= vwidth then
1004                 local dy, n, iy = dy, 0, 0
1005                 if kind == 100 or kind == 103 then -- todo: gleaders
1006                     local ady = abs(ody - dy)
1007                     local ndy = math.ceil(ady / hd) * hd
1008                     local diff = ndy - ady
1009                     n = math.floor((vwidth-diff) / hd)
1010                     dy = dy + (b2u and diff or -diff)
1011                 else
1012                     n = math.floor(vwidth / hd)
1013                     if kind == 101 then
1014                         local side = vwidth % hd / 2
1015                         dy = dy + (b2u and side or -side)
1016                     elseif kind == 102 then
1017                         iy = vwidth % hd / (n+1)
1018                         dy = dy + (b2u and iy or -iy)
1019                     end
1020                 end
1021                 dy = dy + (b2u and curr.depth or -curr.height)/factor
1022                 hd = b2u and hd or -hd
1023                 iy = b2u and iy or -iy
1024                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1025                 for i=1,n do
1026                     res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1027                     dy = dy + hd + iy
1028                 end
1029             end
1030         end
1031     end
1032     dy = dy + (b2u and vwidth or -vwidth)
1033 elseif curr.id == node.id"kern" then
1034     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1035 elseif curr.id == node.id"vlist" then
1036     dy = dy + (b2u and curr.depth or -curr.height)/factor

```

```

1037     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1038     dy = dy + (b2u and curr.height or -curr.depth)/factor
1039 elseif curr.id == node.id"hlist" then
1040     dy = dy + (b2u and curr.depth or -curr.height)/factor
1041     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1042     dy = dy + (b2u and curr.height or -curr.depth)/factor
1043 end
1044 curr = node.getnext(curr)
1045 end
1046 return res
1047 end
1048 function outline_horz (res, box, curr, xshift, yshift, discwd)
1049 local r2l = box.dir == "TRT"
1050 local dx = r2l and (discwd or box.width/factor) or 0
1051 local dirs = { { dir = r2l, dx = dx } }
1052 while curr do
1053     if curr.id == node.id"dir" then
1054         local sign, dir = curr.dir:match"(.)(...)"
1055         local level, newdir = curr.level, r2l
1056         if sign == "+" then
1057             newdir = dir == "TRT"
1058             if r2l ~= newdir then
1059                 local n = node.getnext(curr)
1060                 while n do
1061                     if n.id == node.id"dir" and n.level+1 == level then break end
1062                     n = node.getnext(n)
1063                 end
1064                 n = n or node.tail(curr)
1065                 dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1066             end
1067             dirs[level] = { dir = r2l, dx = dx }
1068         else
1069             local level = level + 1
1070             newdir = dirs[level].dir
1071             if r2l ~= newdir then
1072                 dx = dirs[level].dx
1073             end
1074         end
1075         r2l = newdir
1076     elseif curr.char and curr.font and curr.font > 0 then
1077         local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1078         local gid = ft.characters[curr.char].index or curr.char
1079         local scale = ft.size / factor / 1000
1080         local slant = (ft.slant or 0)/1000
1081         local extend = (ft.extend or 1000)/1000
1082         local squeeze = (ft.squeeze or 1000)/1000
1083         local expand = 1 + (curr.expansion_factor or 0)/1000000
1084         local xscale = scale * extend * expand
1085         local yscale = scale * squeeze

```

```

1086 dx = dx - (r2l and curr.width/factor*expand or 0)
1087 local xpos = dx + xshift + (curr.xoffset or 0)/factor
1088 local ypos = yshift + (curr.yoffset or 0)/factor
1089 local vertical = ""
1090 if ft.shared and ft.shared.features.vertical then -- luatexko
1091     vertical = "rotated 90"
1092     local data = ft.characters[curr.char] or { }
1093     if ft.hb then
1094         local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1095         local charraise = ft.shared.features.charraise or 0
1096         if type(charraise) == "string" then
1097             local n, u = charraise:match"(.+)(e[mx])"
1098             if n and u then
1099                 u = u == "em" and ft.parameters.quad or ft.parameters.x_height
1100                 charraise = (n * u)/factor
1101             else
1102                 charraise = tex.sp(charraise)/factor
1103             end
1104         end
1105         xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1106     else
1107         local cmds = data.commands or { {0,0}, {0,0} }
1108         local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1109         xpos, ypos = xpos + hoff, ypos + voff
1110     end
1111 end
1112 local image
1113 if ft.format == "opentype" or ft.format == "truetype" then
1114     image = luamplib.glyph(curr.font, gid)
1115 else
1116     local name, scale = ft.name, 1
1117     local vf = font.read_vf(name, ft.size)
1118     if vf and vf.characters[gid] then
1119         local cmds = vf.characters[gid].commands or {}
1120         for _,v in ipairs(cmds) do
1121             if v[1] == "char" then
1122                 gid = v[2]
1123             elseif v[1] == "font" and vf.fonts[v[2]] then
1124                 name = vf.fonts[v[2]].name
1125                 scale = vf.fonts[v[2]].size / ft.size
1126             end
1127         end
1128     end
1129     image = format("glyph %s of %q scaled %f", gid, name, scale)
1130 end
1131 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1132     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1133 dx = dx + (r2l and 0 or curr.width/factor*expand)
1134 elseif curr.replace then

```

```

1135     local width = node.dimensions(curr.replace)/factor
1136     dx = dx - (r2l and width or 0)
1137     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1138     dx = dx + (r2l and 0 or width)
1139 elseif curr.id == node.id"rule" then
1140     local wd, ht, dp = getrulemetric(box, curr, true)
1141     if wd ~= 0 then
1142         local hd = ht + dp
1143         dx = dx - (r2l and wd or 0)
1144         if hd ~= 0 and curr.subtype == 0 then
1145             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1146         end
1147         dx = dx + (r2l and 0 or wd)
1148     end
1149 elseif curr.id == node.id"glue" then
1150     local width = node.effective_glue(curr, box)/factor
1151     dx = dx - (r2l and width or 0)
1152     if curr.leader then
1153         local curr, kind = curr.leader, curr.subtype
1154         if curr.id == node.id"rule" then
1155             local wd, ht, dp = getrulemetric(box, curr, true)
1156             local hd = ht + dp
1157             if hd ~= 0 then
1158                 wd = width
1159                 if wd ~= 0 and curr.subtype == 0 then
1160                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1161                 end
1162             end
1163         elseif curr.head then
1164             local wd = curr.width/factor
1165             if wd <= width then
1166                 local dx = r2l and dx+width or dx
1167                 local n, ix = 0, 0
1168                 if kind == 100 or kind == 103 then -- todo: gleaders
1169                     local adx = abs(dx-dirs[1].dx)
1170                     local ndx = math.ceil(adx / wd) * wd
1171                     local diff = ndx - adx
1172                     n = math.floor((width-diff) / wd)
1173                     dx = dx + (r2l and -diff-wd or diff)
1174                 else
1175                     n = math.floor(width / wd)
1176                     if kind == 101 then
1177                         local side = width % wd / 2
1178                         dx = dx + (r2l and -side-wd or side)
1179                     elseif kind == 102 then
1180                         ix = width % wd / (n+1)
1181                         dx = dx + (r2l and -ix-wd or ix)
1182                     end
1183                 end
1184             end
1185         end
1186     end

```



```

1184         wd = r2l and -wd or wd
1185         ix = r2l and -ix or ix
1186         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1187         for i=1,n do
1188             res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1189             dx = dx + wd + ix
1190         end
1191     end
1192 end
1193 end
1194 dx = dx + (r2l and 0 or width)
1195 elseif curr.id == node.id"kern" then
1196     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1197 elseif curr.id == node.id"math" then
1198     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1199 elseif curr.id == node.id"vlist" then
1200     dx = dx - (r2l and curr.width/factor or 0)
1201     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1202     dx = dx + (r2l and 0 or curr.width/factor)
1203 elseif curr.id == node.id"hlist" then
1204     dx = dx - (r2l and curr.width/factor or 0)
1205     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1206     dx = dx + (r2l and 0 or curr.width/factor)
1207 end
1208 curr = node.getnext(curr)
1209 end
1210 return res
1211 end
1212 function luamplib.outlinetext (text)
1213     local fmt = process_tex_text(text)
1214     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1215     local box = texgetbox(id)
1216     local res = outline_horz({ }, box, box.head, 0, 0)
1217     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1218     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1219 end
1220

```

lua functions for mplib(uc)substring ... of ...

```

1221 function luamplib.getunicodegraphemes (s)
1222     local t = { }
1223     local graphemes = require'lua-uni-graphemes'
1224     for _, _, c in graphemes.graphemes(s) do
1225         table.insert(t, c)
1226     end
1227     return t
1228 end
1229 function luamplib.unicodesubstring (s,b,e,grph)
1230     local tt, t, step = { }

```

```

1231 if grph then
1232   t = luamplib.getunicodegraphemes(s)
1233 else
1234   t = { }
1235   for _, c in utf8.codes(s) do
1236     table.insert(t, utf8.char(c))
1237   end
1238 end
1239 if b <= e then
1240   b, step = b+1, 1
1241 else
1242   e, step = e+1, -1
1243 end
1244 for i = b, e, step do
1245   table.insert(tt, t[i])
1246 end
1247 s = table.concat(tt):gsub("'", "'&ditto'")
1248 return string.format("%s", s)
1249 end
1250

```

Our METAPOST preambles

```

1251 luamplib.preambles = {
1252   mplibcode = [[
1253 texscriptmode := 2;
1254 def rawtexttext primary t = runscript("luamplibtext{"&t"}") enddef;
1255 def mplibcolor primary t = runscript("luamplibcolor{"&t"}") enddef;
1256 def mplibdimen primary t = runscript("luamplibdimen{"&t"}") enddef;
1257 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t"}") enddef;
1258 if known context_mlib:
1259   defaultfont := "cmitt10";
1260   let infont = normalinfont;
1261   let fontsize = normalfontsize;
1262   vardef thelabel@#(expr p,z) =
1263     if string p :
1264       thelabel@#(p infont defaultfont scaled defaultscale,z)
1265     else :
1266       p shifted (z + labeloffset*mfun_laboff@# -
1267         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1268         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1269     fi
1270   enddef;
1271 else:
1272   vardef texttext@# primary t = rawtexttext (t) enddef;
1273   def message expr t =
1274     if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1275   enddef;
1276   def withtransparency (expr a, t) =
1277     withprescript "tr_alternative=" & if numeric a: decimal fi a

```

```

1278   withprescript "tr_transparency=" & decimal t
1279   enddef;
1280   vardef ddecimal primary p =
1281     decimal xpart p & " " & decimal ypart p
1282   enddef;
1283   vardef boundingbox primary p =
1284     if (path p) or (picture p) :
1285       llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1286     else :
1287       origin
1288     fi -- cycle
1289   enddef;
1290 fi
1291 def resolvedcolor(expr s) =
1292   runscript("return luamplib.shadecolor('"& s &"')")
1293 enddef;
1294 def colordecimals primary c =
1295   if cmykcolor c:
1296     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1297     decimal yellowpart c & ":" & decimal blackpart c
1298   elseif rgbcolor c:
1299     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1300   elseif string c:
1301     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1302   else:
1303     decimal c
1304   fi
1305 enddef;
1306 def externalfigure primary filename =
1307   draw rawtexttext("\includegraphics{"& filename &"}")
1308 enddef;
1309 def TEX = texttext enddef;
1310 def mplibtexcolor primary c =
1311   runscript("return luamplib.gettexcolor('"& c &"')")
1312 enddef;
1313 def mplibrgbtexcolor primary c =
1314   runscript("return luamplib.gettexcolor('"& c &"', 'rgb')")
1315 enddef;
1316 def mplibgraphicstext primary t =
1317   begingroup;
1318   mplibgraphicstext_ (t)
1319 enddef;
1320 def mplibgraphicstext_ (expr t) text rest =
1321   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor,
1322   fb, fc, dc, graphicstextpic, alsoordoublepath;
1323   picture graphicstextpic; graphicstextpic := nullpicture;
1324   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1325   let scale = scaled;
1326   def fakebold primary c = hide(fb:=c;) enddef;

```

```

1327 def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1328 def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1329 let withfillcolor = fillcolor; let withdrawcolor = drawcolor;
1330 def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1331 addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1332 def fakebold primary c = enddef;
1333 let fillcolor = fakebold; let drawcolor = fakebold;
1334 let withfillcolor = fillcolor; let withdrawcolor = drawcolor;
1335 image(draw runscript("return luamplib.graphictext([==["&t&"]===],"
1336   & decimal fb &","& fc &","& dc &")) rest;))
1337 endgroup;
1338 enddef;
1339 def mplibglyph expr c of f =
1340   runscript (
1341     "return luamplib.glyph('"
1342       & if numeric f: decimal fi f
1343       & "'',"
1344       & if numeric c: decimal fi c
1345       & "')"
1346   )
1347 enddef;
1348 def mplibdrawglyph expr g =
1349   draw image(
1350     save i; numeric i; i:=0;
1351     for item within g:
1352       i := i+1;
1353       fill pathpart item
1354       if i < length g: withpostscript "collect" fi;
1355     endfor
1356   )
1357 enddef;
1358 def mplib_do_outline_text_set_b (text f) (text d) text r =
1359   def mplib_do_outline_options_f = f enddef;
1360   def mplib_do_outline_options_d = d enddef;
1361   def mplib_do_outline_options_r = r enddef;
1362 enddef;
1363 def mplib_do_outline_text_set_f (text f) text r =
1364   def mplib_do_outline_options_f = f enddef;
1365   def mplib_do_outline_options_r = r enddef;
1366 enddef;
1367 def mplib_do_outline_text_set_u (text f) text r =
1368   def mplib_do_outline_options_f = f enddef;
1369 enddef;
1370 def mplib_do_outline_text_set_d (text d) text r =
1371   def mplib_do_outline_options_d = d enddef;
1372   def mplib_do_outline_options_r = r enddef;
1373 enddef;
1374 def mplib_do_outline_text_set_r (text d) (text f) text r =
1375   def mplib_do_outline_options_d = d enddef;

```

```

1376 def mplib_do_outline_options_f = f enddef;
1377 def mplib_do_outline_options_r = r enddef;
1378 enddef;
1379 def mplib_do_outline_text_set_n text r =
1380   def mplib_do_outline_options_r = r enddef;
1381 enddef;
1382 def mplib_do_outline_text_set_p = enddef;
1383 def mplib_fill_outline_text =
1384   for n=1 upto mpliboutlinenum:
1385     i:=0;
1386     for item within mpliboutlinepic[n]:
1387       i:=i+1;
1388       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1389       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1390     endfor
1391   endfor
1392 enddef;
1393 def mplib_draw_outline_text =
1394   for n=1 upto mpliboutlinenum:
1395     for item within mpliboutlinepic[n]:
1396       draw pathpart item mplib_do_outline_options_d;
1397     endfor
1398   endfor
1399 enddef;
1400 def mplib_filldraw_outline_text =
1401   for n=1 upto mpliboutlinenum:
1402     i:=0;
1403     for item within mpliboutlinepic[n]:
1404       i:=i+1;
1405       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1406         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1407       else:
1408         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1409       fi
1410     endfor
1411   endfor
1412 enddef;
1413 vardef mpliboutlinetext@# (expr t) text rest =
1414   save kind; string kind; kind := str @#;
1415   save i; numeric i;
1416   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1417   def mplib_do_outline_options_d = enddef;
1418   def mplib_do_outline_options_f = enddef;
1419   def mplib_do_outline_options_r = enddef;
1420   runscript("return luamplib.outlinetext[===["&t&"]===]");
1421   image ( addto currentpicture also image (
1422     if kind = "f":
1423       mplib_do_outline_text_set_f rest;
1424       mplib_fill_outline_text;

```

```

1425     elseif kind = "d":
1426         mplib_do_outline_text_set_d rest;
1427         mplib_draw_outline_text;
1428     elseif kind = "b":
1429         mplib_do_outline_text_set_b rest;
1430         mplib_fill_outline_text;
1431         mplib_draw_outline_text;
1432     elseif kind = "u":
1433         mplib_do_outline_text_set_u rest;
1434         mplib_filldraw_outline_text;
1435     elseif kind = "r":
1436         mplib_do_outline_text_set_r rest;
1437         mplib_draw_outline_text;
1438         mplib_fill_outline_text;
1439     elseif kind = "p":
1440         mplib_do_outline_text_set_p;
1441         mplib_draw_outline_text;
1442     else:
1443         mplib_do_outline_text_set_n rest;
1444         mplib_fill_outline_text;
1445     fi;
1446 ) mplib_do_outline_options_r; )
1447 enddef ;
1448 def withmppattern primary p =
1449     withprescript "mplibpattern=" & if numeric p: decimal fi p
1450 enddef;
1451 primarydef t withpattern p =
1452     image(
1453         if cycle t:
1454             fill
1455         else:
1456             draw
1457         fi
1458         t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1459 enddef;
1460 vardef mplibtransformmatrix (text e) =
1461     save t; transform t;
1462     t = identity e;
1463     runscript("luamplib.transformmatrix = {"
1464         & decimal xpart t & ","
1465         & decimal ypart t & ","
1466         & decimal xpart t & ","
1467         & decimal ypart t & ","
1468         & decimal xpart t & ","
1469         & decimal ypart t & ","
1470         & "}");
1471 enddef;
1472 primarydef p withfademethod s =
1473     if picture p:

```

```

1474   image(
1475       draw p;
1476       draw center p withprescript "mplibfadestate=stop";
1477   )
1478   else:
1479       p withprescript "mplibfadestate=stop"
1480   fi
1481   withprescript "mplibfadetype=" & s
1482   withprescript "mplibfadebbox=" &
1483       decimal (xpart llcorner p -1/4) & ":" &
1484       decimal (ypart llcorner p -1/4) & ":" &
1485       decimal (xpart urcorner p +1/4) & ":" &
1486       decimal (ypart urcorner p +1/4)
1487   enddef;
1488   def withfadeopacity (expr a,b) =
1489       withprescript "mplibfadeopacity=" &
1490           decimal a & ":" &
1491           decimal b
1492   enddef;
1493   def withfadevector (expr a,b) =
1494       withprescript "mplibfadevector=" &
1495           decimal xpart a & ":" &
1496           decimal ypart a & ":" &
1497           decimal xpart b & ":" &
1498           decimal ypart b
1499   enddef;
1500   let withfadecenter = withfadevector;
1501   def withfaderadius (expr a,b) =
1502       withprescript "mplibfaderadius=" &
1503           decimal a & ":" &
1504           decimal b
1505   enddef;
1506   def withfadebbox (expr a,b) =
1507       withprescript "mplibfadebbox=" &
1508           decimal xpart a & ":" &
1509           decimal ypart a & ":" &
1510           decimal xpart b & ":" &
1511           decimal ypart b
1512   enddef;
1513   primarydef p asgroup s =
1514       image(
1515           draw center p
1516           withprescript "mplibgroupbbox=" &
1517               decimal (xpart llcorner p -1/4) & ":" &
1518               decimal (ypart llcorner p -1/4) & ":" &
1519               decimal (xpart urcorner p +1/4) & ":" &
1520               decimal (ypart urcorner p +1/4)
1521           withprescript "gr_state=start"
1522           withprescript "gr_type=" & s;

```

```

1523     draw p;
1524     draw center p withprescript "gr_state=stop";
1525 )
1526 enddef;
1527 def withgroupbbox (expr a,b) =
1528   withprescript "mplibgroupbbox=" &
1529     decimal xpart a & ":" &
1530     decimal ypart a & ":" &
1531     decimal xpart b & ":" &
1532     decimal ypart b
1533 enddef;
1534 def withgroupname expr s =
1535   withprescript "mplibgroupname=" & s
1536 enddef;
1537 def usemplibgroup primary s =
1538   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s &"\endcsname}")
1539   shifted runscript("return luamplib.trgroupshifts['' & s & ''"]")
1540 enddef;
1541 path    mplib_shade_path ;
1542 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1543 numeric mplib_shade_fx, mplib_shade_fy ;
1544 numeric mplib_shade_lx, mplib_shade_ly ;
1545 numeric mplib_shade_nx, mplib_shade_ny ;
1546 numeric mplib_shade_dx, mplib_shade_dy ;
1547 numeric mplib_shade_tx, mplib_shade_ty ;
1548 primarydef p withshadingmethod m =
1549   p
1550   if picture p :
1551     withprescript "sh_operand_type=picture"
1552     if textual p:
1553       withprescript "sh_transform=no"
1554       mplib_with_shade_method (boundingbox p, m)
1555     else:
1556       withprescript "sh_transform=yes"
1557       mplib_with_shade_method (pathpart p, m)
1558     fi
1559   else :
1560     withprescript "sh_transform=yes"
1561     mplib_with_shade_method (p, m)
1562   fi
1563 enddef;
1564 def mplib_with_shade_method (expr p, m) =
1565   hide(mplib_with_shade_method_analyze(p))
1566   withprescript "sh_domain=0 1"
1567   withprescript "sh_color=into"
1568   withprescript "sh_color_a=" & colordecimals white
1569   withprescript "sh_color_b=" & colordecimals black
1570   withprescript "sh_first=" & ddecimal point 0 of p
1571   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)

```



```

1572 withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1573 if m = "linear" :
1574   withprescript "sh_type=linear"
1575   withprescript "sh_factor=1"
1576   withprescript "sh_center_a=" & ddecimal llcorner p
1577   withprescript "sh_center_b=" & ddecimal urcorner p
1578 else :
1579   withprescript "sh_type=circular"
1580   withprescript "sh_factor=1.2"
1581   withprescript "sh_center_a=" & ddecimal center p
1582   withprescript "sh_center_b=" & ddecimal center p
1583   withprescript "sh_radius_a=" & decimal 0
1584   withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1585 fi
1586 enddef;
1587 def mplib_with_shade_method_analyze(expr p) =
1588   mplib_shade_path := p ;
1589   mplib_shade_step := 1 ;
1590   mplib_shade_fx := xpart point 0 of p ;
1591   mplib_shade_fy := ypart point 0 of p ;
1592   mplib_shade_lx := mplib_shade_fx ;
1593   mplib_shade_ly := mplib_shade_fy ;
1594   mplib_shade_nx := 0 ;
1595   mplib_shade_ny := 0 ;
1596   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1597   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1598   for i=1 upto length(p) :
1599     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1600     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1601     if mplib_shade_tx > mplib_shade_dx :
1602       mplib_shade_nx := i + 1 ;
1603       mplib_shade_lx := xpart point i of p ;
1604       mplib_shade_dx := mplib_shade_tx ;
1605     fi ;
1606     if mplib_shade_ty > mplib_shade_dy :
1607       mplib_shade_ny := i + 1 ;
1608       mplib_shade_ly := ypart point i of p ;
1609       mplib_shade_dy := mplib_shade_ty ;
1610     fi ;
1611   endfor ;
1612 enddef;
1613 vardef mplib_max_radius(expr p) =
1614   max (
1615     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1616     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1617     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1618     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1619   )
1620 enddef;

```

```

1621 def withshadingstep (text t) =
1622   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1623   withprescript "sh_step=" & decimal mplib_shade_step
1624   t
1625 enddef;
1626 def withshadingradius expr a =
1627   withprescript "sh_radius_a=" & decimal (xpart a)
1628   withprescript "sh_radius_b=" & decimal (ypart a)
1629 enddef;
1630 def withshadingorigin expr a =
1631   withprescript "sh_center_a=" & ddecimal a
1632   withprescript "sh_center_b=" & ddecimal a
1633 enddef;
1634 def withshadingvector expr a =
1635   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1636   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1637 enddef;
1638 def withshadingdirection expr a =
1639   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1640   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1641 enddef;
1642 def withshadingtransform expr a =
1643   withprescript "sh_transform=" & a
1644 enddef;
1645 def withshadingcenter expr a =
1646   withprescript "sh_center_a=" & ddecimal (
1647     center mplib_shade_path shifted (
1648       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1649       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1650     )
1651   )
1652 enddef;
1653 def withshadingdomain expr d =
1654   withprescript "sh_domain=" & ddecimal d
1655 enddef;
1656 def withshadingfactor expr f =
1657   withprescript "sh_factor=" & decimal f
1658 enddef;
1659 def withshadingfraction expr a =
1660   if mplib_shade_step > 0 :
1661     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1662   fi
1663 enddef;
1664 def withshadingcolors (expr a, b) =
1665   if mplib_shade_step > 0 :
1666     withprescript "sh_color=into"
1667     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1668     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1669   else :

```

```

1670     withprescript "sh_color=into"
1671     withprescript "sh_color_a=" & colordecimals a
1672     withprescript "sh_color_b=" & colordecimals b
1673   fi
1674 enddef;
1675 def mpliblength primary t =
1676   runscript("return utf8.len[===[" & t & "]===")
1677 enddef;
1678 def mplibsubstring expr p of t =
1679   runscript("return luamplib.unicodesubstring([===[" & t & "]===")
1680     & decimal xpart p & ","
1681     & decimal ypart p & ")")
1682 enddef;
1683 def mplibuclength primary t =
1684   runscript("return #luamplib.getunicodegraphemes[===[" & t & "]===")
1685 enddef;
1686 def mplibucsubstring expr p of t =
1687   runscript("return luamplib.unicodesubstring([===[" & t & "]===")
1688     & decimal xpart p & ","
1689     & decimal ypart p & ",true)")
1690 enddef;
1691 ]],
1692 legacyverbatimtex = [[
1693 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1694 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1695 let VerbatimTeX = specialVerbatimTeX;
1696 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1697   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1698 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1699   "runscript(" &ditto&
1700   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1701   "luamplib.in_the_fig=false" &ditto& ");";
1702 ]],
1703 texttextlabel = [[
1704 let luampliboriginalinfont = infont;
1705 primarydef s infont f =
1706   if (s < char 32)
1707     or (s = char 35) % #
1708     or (s = char 36) % $
1709     or (s = char 37) % %
1710     or (s = char 38) % &
1711     or (s = char 92) % \
1712     or (s = char 94) % ^
1713     or (s = char 95) % _
1714     or (s = char 123) % {
1715     or (s = char 125) % }
1716     or (s = char 126) % ~
1717     or (s = char 127) :
1718     s luampliboriginalinfont f

```

```

1719 else :
1720   rawtexttext(s)
1721 fi
1722 enddef;
1723 def fontsize expr f =
1724   begingroup
1725   save size; numeric size;
1726   size := mplibdimen("1em");
1727   if size = 0: 10pt else: size fi
1728 endgroup
1729 enddef;
1730 ]],
1731 }
1732

```

When `\mplibverbatim` is enabled, do not expand `mplibcode` data.

```

1733 luamplib.verbatiminput = false
1734 luamplib.everymplib     = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
1735 luamplib.everyendmplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
1736 function luamplib.process_mplibcode (data, instancename)
1737   texboxes.localid = 4096

```

This is needed for legacy behavior

```

1738 if luamplib.legacyverbatim then
1739   luamplib.figid, tex_code_pre_mplib = 1, {}
1740 end
1741 local everymplib   = luamplib.everymplib[instancename]
1742 local everyendmplib = luamplib.everyendmplib[instancename]
1743 data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
1744 :gsub("\r", "\n")

```

These five lines are needed for `mplibverbatim` mode.

```

1745 if luamplib.verbatiminput then
1746   data = data:gsub("\mpcolor%s+(.-%b{ })", "mplibcolor(\\"%1\\")")
1747   :gsub("\mpdim%s+(%b{ })", "mplibdimen(\\"%1\\")")
1748   :gsub("\mpdim%s+(\\%a+)", "mplibdimen(\\"%1\\")")
1749   :gsub(btex_etex, "btex %1 etex ")
1750   :gsub(verbatimetex_etex, "verbatimetex %1 etex;")
1751 else

```

If not `mplibverbatim`, expand `mplibcode` data, so that users can use \TeX codes in it. It has turned out that no comment sign is allowed. However, we do not expand `btex ... etex`, `verbatimetex ... etex`, and string expressions.

```

1752   local t = { } -- to store btex, verbatimetex, string
1753   data = data:gsub(btex_etex, function(str)
1754     t[#t+1] = str
1755     return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
1756   end)
1757   :gsub(verbatimetex_etex, function(str)
1758     t[#t+1] = str

```

```

1759     return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
1760 end)
1761 :gsub('"(.)"', function(str)
1762     t[#t+1] = str
1763     return format("'\\unexpanded{!l!u!a!%s!m!p!l!}'", #t)
1764 end)
1765 :gsub("\\%", "\\0PerCent\\0")
1766 :gsub("%%.-\\n", "\\n")
1767 :gsub("%zPerCent%z", "\\%")
1768 run_tex_code(format("\\mplibtmp toks\\expandafter{\\expanded{%s}}", data))
1769 data = texgettoks"mplibtmp toks"

```

Next line to address issue #55

```

1770 :gsub("##", "#")
1771 :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
1772 end
1773 process(data, instanceName)
1774 end
1775

```

For parsing prescript materials.

```

1776 local function script2table(s)
1777     local t = {}
1778     for _,i in ipairs(s:explode("\\13+")) do
1779         local k,v = i:match("(.)=(.*)") -- v may contain = or empty.
1780         if k and v and k ~= "" and not t[k] then
1781             t[k] = v
1782         end
1783     end
1784     return t
1785 end
1786

```

pdf literals will be stored in figcontents table, and written to pdf in one go at the end of the flushing figure. Subtable post is for the legacy behavior.

```

1787 local figcontents = { post = { } }
1788 local function put2output(a,...)
1789     figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
1790 end
1791 local function pdf_startfigure(n,llx,lly,urx,ury)
1792     put2output("\\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
1793 end
1794 local function pdf_stopfigure()
1795     put2output("\\mplibstoptoPDF")
1796 end

```

tex.sprint with catcode regime -2, as sometimes # gets doubled in the argument of pdf literal.

```

1797 local function pdf_literalcode (...)
1798     put2output{-2, (format(...) :gsub(decimals,rmzeros)) }

```

```

1799 end
1800 local start_pdf_code = pdfmode
1801 and function() pdf_literalcode"q" end
1802 or function() put2output"\special{pdf:bcontent}" end
1803 local stop_pdf_code = pdfmode
1804 and function() pdf_literalcode"Q" end
1805 or function() put2output"\special{pdf:econtent}" end
1806

```

Now we process hboxes created from `btex ... etex` or `texttext(...)` or `TEX(...)`, all being the same internally.

```

1807 local function put_tex_boxes (object,prescript)
1808   local box = prescript.mplibtexboxid:explode":"
1809   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
1810   if n and tw and th then
1811     local op = object.path
1812     local first, second, fourth = op[1], op[2], op[4]
1813     local tx, ty = first.x_coord, first.y_coord
1814     local sx, rx, ry, sy = 1, 0, 0, 1
1815     if tw ~= 0 then
1816       sx = (second.x_coord - tx)/tw
1817       rx = (second.y_coord - ty)/tw
1818       if sx == 0 then sx = 0.00001 end
1819     end
1820     if th ~= 0 then
1821       sy = (fourth.y_coord - ty)/th
1822       ry = (fourth.x_coord - tx)/th
1823       if sy == 0 then sy = 0.00001 end
1824     end
1825     start_pdf_code()
1826     pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
1827     put2output("\mplibputtextbox{%i}",n)
1828     stop_pdf_code()
1829   end
1830 end
1831

```

Colors

```

1832 local prev_override_color
1833 local function do_preobj_CR(object,prescript)
1834   if object.postscript == "collect" then return end
1835   local override = prescript and prescript.mpliboverridecolor
1836   if override then
1837     if pdfmode then
1838       pdf_literalcode(override)
1839       override = nil
1840     else
1841       put2output("\special{%s}",override)
1842       prev_override_color = override
1843     end

```

```

1844 else
1845     local cs = object.color
1846     if cs and #cs > 0 then
1847         pdf_literalcode(luamplib.colorconverter(cs))
1848         prev_override_color = nil
1849     elseif not pdfmode then
1850         override = prev_override_color
1851         if override then
1852             put2output("\\special{%s}",override)
1853         end
1854     end
1855 end
1856 return override
1857 end
1858

```

For transparency and shading

```

1859 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
1860 local pdfobjs, pdfetcs = {}, {}
1861 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
1862 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
1863 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
1864 local function update_pdfobjs (os, stream)
1865     local key = os
1866     if stream then key = key..stream end
1867     local on = key and pdfobjs[key]
1868     if on then
1869         return on,false
1870     end
1871     if pdfmode then
1872         if stream then
1873             on = pdf.immediateobj("stream",stream,os)
1874         elseif os then
1875             on = pdf.immediateobj(os)
1876         else
1877             on = pdf.reserveobj()
1878         end
1879     else
1880         on = pdfetcs.cnt or 1
1881         if stream then
1882             texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
1883         elseif os then
1884             texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
1885         else
1886             texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
1887         end
1888         pdfetcs.cnt = on + 1
1889     end
1890     if key then

```

```

1891 pdfobjs[key] = on
1892 end
1893 return on,true
1894 end
1895 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
1896 if pdfmode then
1897 pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
1898 local getpagers = pdfetcs.getpagers
1899 local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
1900 local initialize_resources = function (name)
1901 local tabname = format("%s_res",name)
1902 pdfetcs[tabname] = { }
1903 if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
1904 local obj = pdf.reserveobj()
1905 setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
1906 luatexbase.add_to_callback("finish_pdffile", function()
1907 pdf.immediateobj(obj, format("<<s>>", tableconcat(pdfetcs[tabname])))
1908 end,
1909 format("luamplib.%s.finish_pdffile",name))
1910 end
1911 end
1912 pdfetcs.fallback_update_resources = function (name, res)
1913 local tabname = format("%s_res",name)
1914 if not pdfetcs[tabname] then
1915 initialize_resources(name)
1916 end
1917 if luatexbase.callbacktypes.finish_pdffile then
1918 local t = pdfetcs[tabname]
1919 t[#t+1] = res
1920 else
1921 local tpr, n = getpagers() or "", 0
1922 tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
1923 if n == 0 then
1924 tpr = format("%s/%s<<s>>", tpr, name, res)
1925 end
1926 setpagers(tpr)
1927 end
1928 end
1929 else
1930 texsprint {
1931 "\\luamplibatfirstshipout{",
1932 "\\special{pdf:obj @MPLibTr<<>>}",
1933 "\\special{pdf:obj @MPLibSh<<>>}",
1934 "\\special{pdf:obj @MPLibCS<<>>}",
1935 "\\special{pdf:obj @MPLibPt<<>>}}",
1936 }
1937 pdfetcs.resadded = { }
1938 pdfetcs.fallback_update_resources = function (name,res,obj)
1939 texsprint{"\\special{pdf:put ", obj, " <<", res, ">>"}"}

```



```

1940   if not pdfetcs.resadded[name] then
1941       texsprint{"\\luampliateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
1942       pdfetcs.resadded[name] = obj
1943   end
1944 end
1945 end
1946

```

Transparency

```

1947 local transparency_modes = { [0] = "Normal",
1948   "Normal",      "Multiply",    "Screen",      "Overlay",
1949   "SoftLight",   "HardLight",   "ColorDodge",  "ColorBurn",
1950   "Darken",      "Lighten",     "Difference",  "Exclusion",
1951   "Hue",         "Saturation", "Color",      "Luminosity",
1952   "Compatible",
1953   normal        = "Normal",    multiply      = "Multiply",    screen       = "Screen",
1954   overlay       = "Overlay",    softlight    = "SoftLight",   hardlight    = "HardLight",
1955   colordodge    = "ColorDodge", colorburn     = "ColorBurn",   darken       = "Darken",
1956   lighten      = "Lighten",     difference    = "Difference",  exclusion    = "Exclusion",
1957   hue           = "Hue",        saturation    = "Saturation",  color        = "Color",
1958   luminosity    = "Luminosity", compatible    = "Compatible",
1959 }
1960 local function add_extgs_resources (on, new)
1961   local key = format("MPlibTr%s", on)
1962   if new then
1963     local val = format(pdfetcs.resfmt, on)
1964     if pdfmanagement then
1965       texsprint {
1966         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
1967       }
1968     else
1969       local tr = format("/%s %s", key, val)
1970       if is_defined(pdfetcs.pgfgextgs) then
1971         texsprint { "\\csname ", pdfetcs.pgfgextgs, "\\endcsname{" , tr, "}" }
1972       elseif is_defined"TRP@list" then
1973         texsprint(catat11,{
1974           [[\if@files\immediate\write\@auxout{]],
1975           [[\string\g@addto@macro\string\TRP@list{]],
1976           tr,
1977           [[}]\fi]],
1978         })
1979         if not get_macro"TRP@list":find(tr) then
1980           texsprint(catat11,[[\global\TRP@reruntrue]])
1981         end
1982       else
1983         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPlibTr")
1984       end
1985     end
1986 end

```

```

1987 return key
1988 end
1989 local function do_preobj_TR(object,prescript)
1990   if object.postscript == "collect" then return end
1991   local opaq = prescript and prescript.tr_transparency
1992   if opaq then
1993     local key, on, os, new
1994     local mode = prescript.tr_alternative or 1
1995     mode = transparency_modes[tonumber(mode) or mode:lower()]
1996     if not mode then
1997       mode = prescript.tr_alternative
1998       warn("unsupported blend mode: '%s'", mode)
1999     end
2000     opaq = format("%.3f", opaq) :gsub(decimals,rmzeros)
2001     for i,v in ipairs[ {mode,opaq},{ "Normal",1} ] do
2002       os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[2])
2003       on, new = update_pdfobjs(os)
2004       key = add_extgs_resources(on,new)
2005       if i == 1 then
2006         pdf_literalcode("/%s gs",key)
2007       else
2008         return format("/%s gs",key)
2009       end
2010     end
2011   end
2012 end
2013

```

Shading with *metafun* format.

```

2014 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions)
2015   for _,v in ipairs{ca,cb} do
2016     for i,vv in ipairs(v) do
2017       for ii,vvv in ipairs(vv) do
2018         v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2019       end
2020     end
2021   end
2022   local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2023   if steps > 1 then
2024     local list,bounds,encode = { },{ },{ }
2025     for i=1,steps do
2026       if i < steps then
2027         bounds[i] = format("%.3f", fractions[i] or 1)
2028       end
2029       encode[2*i-1] = 0
2030       encode[2*i] = 1
2031       os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
2032       :gsub(decimals,rmzeros)
2033       list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))

```

```

2034     end
2035     os = tableconcat {
2036         "<</FunctionType 3",
2037         format("/Bounds[%s]", tableconcat(bounds, ' ')),
2038         format("/Encode[%s]", tableconcat(encode, ' ')),
2039         format("/Functions[%s]", tableconcat(list, ' ')),
2040         format("/Domain[%s]>>", domain),
2041     } :gsub(decimals,rmzeros)
2042 else
2043     os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
2044     :gsub(decimals,rmzeros)
2045 end
2046 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2047 os = tableconcat {
2048     format("<</ShadingType %i", shtype),
2049     format("/ColorSpace %s", colorspace),
2050     format("/Function %s", objref),
2051     format("/Coords[%s]", coordinates),
2052     "/Extend[true true]/AntiAlias true>>",
2053 } :gsub(decimals,rmzeros)
2054 local on, new = update_pdfobjs(os)
2055 if new then
2056     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2057     if pdfmanagement then
2058         texsprint {
2059             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2060         }
2061     else
2062         local res = format("/%s %s", key, val)
2063         pdfetcs.fallback_update_resources("Shading",res,"@MPlibSh")
2064     end
2065 end
2066 return on
2067 end
2068 local function color_normalize(ca,cb)
2069     if #cb == 1 then
2070         if #ca == 4 then
2071             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2072         else -- #ca = 3
2073             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2074         end
2075     elseif #cb == 3 then -- #ca == 4
2076         cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
2077     end
2078 end
2079 pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2080     run_tex_code({
2081         [[\color_model_new:nnn]],
2082         format("{mplibcolorspace_%s}", names:gsub(",","_")),

```

```

2083     format("{DeviceN}{names={%s}}", names),
2084     [[\edef\mplib_atempa{\pdf_object_ref_last:}]],
2085   }, ccexplat)
2086   local colorspace = get_macro'mplib_atempa'
2087   t[names] = colorspace
2088   return colorspace
2089 end })
2090 local function do_preobj_SH(object,prescript)
2091   local shade_no
2092   local sh_type = prescript and prescript.sh_type
2093   if not sh_type then
2094     return
2095   else
2096     local domain = prescript.sh_domain or "0 1"
2097     local centera = (prescript.sh_center_a or "0 0"):explode()
2098     local centerb = (prescript.sh_center_b or "0 0"):explode()
2099     local transform = prescript.sh_transform == "yes"
2100     local sx,sy,sr,dx,dy = 1,1,1,0,0
2101     if transform then
2102       local first = (prescript.sh_first or "0 0"):explode()
2103       local setx = (prescript.sh_set_x or "0 0"):explode()
2104       local sety = (prescript.sh_set_y or "0 0"):explode()
2105       local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2106       if x ~= 0 and y ~= 0 then
2107         local path = object.path
2108         local path1x = path[1].x_coord
2109         local path1y = path[1].y_coord
2110         local path2x = path[x].x_coord
2111         local path2y = path[y].y_coord
2112         local dxa = path2x - path1x
2113         local dya = path2y - path1y
2114         local dxb = setx[2] - first[1]
2115         local dyb = sety[2] - first[2]
2116         if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2117           sx = dxa / dxb ; if sx < 0 then sx = - sx end
2118           sy = dya / dyb ; if sy < 0 then sy = - sy end
2119           sr = math.sqrt(sx^2 + sy^2)
2120           dx = path1x - sx*first[1]
2121           dy = path1y - sy*first[2]
2122         end
2123       end
2124     end
2125     local ca, cb, colorspace, steps, fractions
2126     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:"" }
2127     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:"" }
2128     steps = tonumber(prescript.sh_step) or 1
2129     if steps > 1 then
2130       fractions = { prescript.sh_fraction_1 or 0 }
2131       for i=2,steps do

```

```

2132     fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2133     ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
2134     cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
2135   end
2136 end
2137 if prescript.mplib_spotcolor then
2138   ca, cb = { }, { }
2139   local names, pos, objref = { }, -1, ""
2140   local script = object.prescript:explode"\13+"
2141   for i=#script,1,-1 do
2142     if script[i]:find"mplib_spotcolor" then
2143       local t, name, value = script[i]:explode"="[2]:explode":"
2144       value, objref, name = t[1], t[2], t[3]
2145       if not names[name] then
2146         pos = pos+1
2147         names[name] = pos
2148         names[#names+1] = name
2149       end
2150       t = { }
2151       for j=1,names[name] do t[#t+1] = 0 end
2152       t[#t+1] = value
2153       tableinsert(#ca == #cb and ca or cb, t)
2154     end
2155   end
2156   for _,t in ipairs{ca,cb} do
2157     for _,tt in ipairs(t) do
2158       for i=1,#names-#tt do tt[#tt+1] = 0 end
2159     end
2160   end
2161   if #names == 1 then
2162     colorspace = objref
2163   else
2164     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2165   end
2166 else
2167   local model = 0
2168   for _,t in ipairs{ca,cb} do
2169     for _,tt in ipairs(t) do
2170       model = model > #tt and model or #tt
2171     end
2172   end
2173   for _,t in ipairs{ca,cb} do
2174     for _,tt in ipairs(t) do
2175       if #tt < model then
2176         color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2177       end
2178     end
2179   end
2180   colorspace = model == 4 and "/DeviceCMYK"

```

```

2181         or model == 3 and "/DeviceRGB"
2182         or model == 1 and "/DeviceGray"
2183         or err"unknown color model"
2184     end
2185     if sh_type == "linear" then
2186         local coordinates = format("%f %f %f %f",
2187             dx + sx*centera[1], dy + sy*centera[2],
2188             dx + sx*centerb[1], dy + sy*centerb[2])
2189         shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions)
2190     elseif sh_type == "circular" then
2191         local factor = prescript.sh_factor or 1
2192         local radiusa = factor * prescript.sh_radius_a
2193         local radiusb = factor * prescript.sh_radius_b
2194         local coordinates = format("%f %f %f %f %f %f",
2195             dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2196             dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2197         shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions)
2198     else
2199         err"unknown shading type"
2200     end
2201 end
2202 return shade_no
2203 end
2204

```

Shading Patterns: much similar to the metafun's shade, but we can apply shading to textual pictures as well as paths.

```

2205 if not pdfmode then
2206     pdfetcs.patternresources = {}
2207 end
2208 local function add_pattern_resources (key, val)
2209     if pdfmanagement then
2210         texsprint {
2211             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2212         }
2213     else
2214         local res = format("/%s %s", key, val)
2215         if is_defined(pdfetcs.pgfpattern) then
2216             texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2217         else
2218             pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2219             if not pdfmode then
2220                 tableinsert(pdfetcs.patternresources, res) -- for gather_resources()
2221             end
2222         end
2223     end
2224 end
2225 function luamplib.dolatelua (on, os)
2226     local h, v = pdf.getpos()

```

```

2227 h = format("%f", h/factor) :gsub(decimals,rmzeros)
2228 v = format("%f", v/factor) :gsub(decimals,rmzeros)
2229 if pdfmode then
2230   pdf.obj(on, format("<<Matrix[1 0 0 1 %s %s]>>", os, h, v))
2231   pdf.refobj(on)
2232 else
2233   local shift = os:explode()
2234   if tonumber(h) ~= tonumber(shift[1]) or tonumber(v) ~= tonumber(shift[2]) then
2235     warn([[Add 'withprescript "sh_matrixshift=%s %s"' to the picture shading]], h, v)
2236   end
2237 end
2238 end
2239 local function do_preobj_shading (object, prescript)
2240   if not prescript or not prescript.sh_operand_type then return end
2241   local on = do_preobj_SH(object, prescript)
2242   local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2243   on = update_pdfobjs()
2244   if pdfmode then
2245     put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(\"on,\"[[\",os,\"]]) }" })
2246   else

```

Why @xpos @ypos do not work properly???

Anyway, this seems to be needed for proper functioning:

```

\pagewidth=\paperwidth
\pageheight=\paperheight
\special{papersize=\the\paperwidth,\the\paperheight}

2247   if is_defined"RecordProperties" then
2248     put2output(tableconcat{
2249       "\\csname tex_savepos:D\\endcsname\\RecordProperties{luamplib/getpos/\",on,\"}{xpos,ypos}\\z
2250       \\special{pdf:put @mplibpdfobj\",on,\" <<\",os,\"/Matrix[1 0 0 1 \\z
2251       \\csname dim_to_decimal_in_bp:n\\endcsname{\\RefProperty{luamplib/getpos/\",on,\"}{xpos}sp} \\z
2252       \\csname dim_to_decimal_in_bp:n\\endcsname{\\RefProperty{luamplib/getpos/\",on,\"}{ypos}sp}\\z
2253       ]>>}\"
2254     })
2255   else
2256     local shift = prescript.sh_matrixshift or "0 0"
2257     texsprint{ "\\special{pdf:put @mplibpdfobj\",on,\" <<\",os,\"/Matrix[1 0 0 1 \",shift,\"]>>}\" }
2258     put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(\"on,\"[[\",shift,\"]]) }" })
2259   end
2260 end
2261 local key, val = format("MPLibPt%s", on), format(pdfetcs.resfmt, on)
2262 add_pattern_resources(key,val)
2263 pdf_literalcode("/Pattern cs/%s scn", key)

```

To avoid possible double execution, once by Pattern gs, once by Sh operator.

```

2264 prescript.sh_type = nil
2265 end
2266

```

Tiling Patterns

```

2267 pdfetcs.patterns = { }
2268 local function gather_resources (optres)
2269   local t, do_pattern = { }, not optres
2270   local names = {"ExtGState", "ColorSpace", "Shading"}
2271   if do_pattern then
2272     names[#names+1] = "Pattern"
2273   end
2274   if pdfmode then
2275     if pdfmanagement then
2276       for _,v in ipairs(names) do
2277         if ltx.__pdf.Page.Resources[v] then
2278           t[#t+1] = format("/%s %s 0 R", v, ltx.pdf.object_id("__pdf/Page/Resources/"..v))
2279         end
2280       end
2281     else
2282       local res = pdfetcs.getpageres() or ""
2283       run_tex_code[["\mplibtmptoks\expandafter{\the\pdfvariable pageresources}"]]
2284       res = res .. texgettoks'mplibtmptoks'
2285       if do_pattern then return res end
2286       res = res:explode"/+"
2287       for _,v in ipairs(res) do
2288         v = v:match"^%s*(.)%s*$"
2289         if not v:find"Pattern" and not optres:find(v) then
2290           t[#t+1] = "/" .. v
2291         end
2292       end
2293     end
2294   else
2295     if pdfmanagement then
2296       for _,v in ipairs(names) do
2297         run_tex_code ({
2298           "\mplibtmptoks\expanded{",
2299           "\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/", v, "}",
2300           "{/", v, " \pdf_object_ref:n{__pdf/Page/Resources/", v, "}}",
2301         }, ccexplat)
2302         t[#t+1] = texgettoks'mplibtmptoks'
2303       end
2304     elseif is_defined(pdfetcs.pgfgextgs) then
2305       run_tex_code ({
2306         "\mplibtmptoks\expanded{",
2307         "\ifpgf@sys@pdf@extgs@exists /ExtGState @pgfgextgs\\fi",
2308         "\ifpgf@sys@pdf@colorspaces@exists /ColorSpace @pgfcolorspaces\\fi",
2309         do_pattern and "\ifpgf@sys@pdf@patterns@exists /Pattern @pgfpatterns \\fi" or "",
2310         "}}",
2311       }, catat11)
2312       t[#t+1] = texgettoks'mplibtmptoks'
2313     if pdfetcs.resadded.Shading then
2314       t[#t+1] = format("/Shading %s", pdfetcs.resadded.Shading)

```



```

2315     end
2316   else
2317     for _,v in ipairs(names) do
2318       local vv = pdfetcs.resadded[v]
2319       if vv then
2320         t[#t+1] = format("/%s %s", v, vv)
2321       end
2322     end
2323   end
2324 end
2325 if do_pattern then return tableconcat(t) end
2326 -- get pattern resources
2327 local mytoks
2328 if pdfmanagement then
2329   run_tex_code ({
2330     "\\mplibmptoks\\expanded{{" ,
2331     "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/Pattern}",
2332     "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/Pattern}}", "}"},
2333   },ccexplat)
2334   mytoks = texgettoks"mplibmptoks"
2335   if not pdfmode then
2336     mytoks = mytoks:gsub("\\str_convert_pdfname:n%s*{(.-)}", "%1") -- why not expanded?
2337   end
2338 elseif is_defined(pdfetcs.pgftextgs) then
2339   if pdfmode then
2340     mytoks = get_macro"pgf@sys@pgf@resource@list@patterns"
2341   else
2342     local tt, abc = {}, get_macro"pgfutil@abc" or ""
2343     for v in abc:gmatch"@pgfpatterns%s*<(.-)>>" do
2344       tt[#tt+1] = v
2345     end
2346     mytoks = tableconcat(tt)
2347   end
2348 else
2349   local tt = pdfmode and pdfetcs.Pattern_res or pdfetcs.patternresources
2350   mytoks = tt and tableconcat(tt)
2351 end
2352 if mytoks and mytoks ~= "" then
2353   t[#t+1] = format("/Pattern<<%s>>",mytoks)
2354 end
2355 return tableconcat(t)
2356 end
2357 function luamplib.registerpattern ( boxid, name, opts )
2358   local box = texgetbox(boxid)
2359   local wd = format("%.3f",box.width/factor)
2360   local hd = format("%.3f", (box.height+box.depth)/factor)
2361   info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2362   if opts.xstep == 0 then opts.xstep = nil end
2363   if opts.ystep == 0 then opts.ystep = nil end

```

```

2364 if opts.colored == nil then
2365   opts.colored = opts.coloured
2366   if opts.colored == nil then
2367     opts.colored = true
2368   end
2369 end
2370 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2371 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2372 if opts.matrix and opts.matrix:find"%a" then
2373   local data = format("mplibtransformmatrix(%s);", opts.matrix)
2374   process(data, "@mplibtransformmatrix")
2375   local t = luamplib.transformmatrix
2376   opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2377   opts.xshift = opts.xshift or format("%f", t[5])
2378   opts.yshift = opts.yshift or format("%f", t[6])
2379 end
2380 local attr = {
2381   "/Type/Pattern",
2382   "/PatternType 1",
2383   format("/PaintType %i", opts.colored and 1 or 2),
2384   "/TilingType 2",
2385   format("/XStep %s", opts.xstep or wd),
2386   format("/YStep %s", opts.ystep or hd),
2387   format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2388 }
2389 local optres = opts.resources or ""
2390 optres = optres .. gather_resources(optres)
2391 local patterns = pdfetcs.patterns
2392 if pdfmode then
2393   if opts.bbox then
2394     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2395   end
2396   attr = tableconcat(attr) :gsub(decimals, rmzeros)
2397   local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2398   patterns[name] = { id = index, colored = opts.colored }
2399 else
2400   local cnt = #patterns + 1
2401   local objname = "@mplibpattern" .. cnt
2402   local metric = format("bbox %s", opts.bbox or format("0 0 %s %s", wd, hd))
2403   texsprint {
2404     "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2405     "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2406     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2407     "\\special{pdf:bcontent}",
2408     "\\special{pdf:bxobj ", objname, " ", metric, "}",
2409     "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2410     "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2411     "\\special{pdf:put @resources <<", optres, ">>}",
2412     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",

```

```

2413     "\\special{pdf:econtent}}",
2414 }
2415 patterns[cnt] = objname
2416 patterns[name] = { id = cnt, colored = opts.colored }
2417 end
2418 end
2419 local function pattern_colorspace (cs)
2420 local on, new = update_pdfobjs(format("/Pattern %s", cs))
2421 if new then
2422     local key, val = format("MPLibCS%i",on), format(pdfetcs.resfmt,on)
2423     if pdfmanagement then
2424         texsprint {
2425             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
2426         }
2427     else
2428         local res = format("/%s %s", key, val)
2429         if is_defined(pdfetcs.pgfcolorspace) then
2430             texsprint { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
2431         else
2432             pdfetcs.fallback_update_resources("ColorSpace",res,"@MPLibCS")
2433         end
2434     end
2435 end
2436 return on
2437 end
2438 local function do_preobj_PAT(object, prescript)
2439 local name = prescript and prescript.mplibpattern
2440 if not name then return end
2441 local patterns = pdfetcs.patterns
2442 local patt = patterns[name]
2443 local index = patt and patt.id or err("cannot get pattern object '%s'", name)
2444 local key = format("MPLibPt%s",index)
2445 if patt.colored then
2446     pdf_literalcode("/Pattern cs /%s scn", key)
2447 else
2448     local color = prescript.mpliboverridecolor
2449     if not color then
2450         local t = object.color
2451         color = t and #t>0 and luamplib.colorconverter(t)
2452     end
2453     if not color then return end
2454     local cs
2455     if color:find" cs " or color:find"@pdf.obj" then
2456         local t = color:explode()
2457         if pdfmode then
2458             cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
2459             color = t[3]
2460         else
2461             cs = t[2]

```

```

2462     color = t[3]:match"%[(.+)%"
2463     end
2464   else
2465     local t = colorsplit(color)
2466     cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
2467     color = tableconcat(t, " ")
2468   end
2469   pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
2470 end
2471 if not patt.done then
2472   local val = pdfmode and format("%s 0 R", index) or patterns[index]
2473   add_pattern_resources(key, val)
2474 end
2475 patt.done = true
2476 end
2477

```

Fading

```

2478 pdfetcs.fading = { }
2479 local function do_preobj_FADE (object, prescript)
2480   local fd_type = prescript and prescript.mplibfadetype
2481   local fd_stop = prescript and prescript.mplibfadestate
2482   if not fd_type then
2483     return fd_stop -- returns "stop" (if picture) or nil
2484   end
2485   local bbox = prescript.mplibfadebbox:explode":""
2486   local dx, dy = -bbox[1], -bbox[2]
2487   local vec = prescript.mplibfadevector; vec = vec and vec:explode":""
2488   if not vec then
2489     if fd_type == "linear" then
2490       vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
2491     else
2492       local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
2493       vec = {centerx, centery, centerx, centery} -- center for both circles
2494     end
2495   end
2496   local coords = { vec[1]+dx, vec[2]+dy, vec[3]+dx, vec[4]+dy }
2497   if fd_type == "linear" then
2498     coords = format("%f %f %f %f", tableunpack(coords))
2499   elseif fd_type == "circular" then
2500     local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
2501     local radius = (prescript.mplibfaderadius or "0:"..math.sqrt(width^2+height^2)/2):explode":""
2502     tableinsert(coords, 3, radius[1])
2503     tableinsert(coords, radius[2])
2504     coords = format("%f %f %f %f %f %f", tableunpack(coords))
2505   else
2506     err("unknown fading method '%s'", fd_type)
2507   end
2508   fd_type = fd_type == "linear" and 2 or 3

```

```

2509 local opa = (prescript.mplibfadeopacity or "1:0"):explode":"
2510 local on, os, new
2511 on = sh_pdfpageresources(fd_type, "0 1", "/DeviceGray", {{opa[1]}}, {{opa[2]}}, coords, 1)
2512 os = format("<</PatternType 2/Shading %s>>", format(pdfetcs.resfmt, on))
2513 on = update_pdfobjs(os)
2514 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
2515 local streamtext = format("q /Pattern cs/MPLibFd%$ scn %$ re f Q", on, bbox)
2516 :gsub(decimals,rmzeros)
2517 os = format("<</Pattern<</MPLibFd%$ %s>>>>", on, format(pdfetcs.resfmt, on))
2518 on = update_pdfobjs(os)
2519 local resources = format(pdfetcs.resfmt, on)
2520 on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
2521 local attr = tableconcat{
2522   "/Subtype/Form",
2523   "/BBox[" .. bbox .. "]",
2524   "/Matrix[1 0 0 1 " .. format("%f %f", -dx, -dy) .. "]",
2525   "/Resources " .. resources,
2526   "/Group " .. format(pdfetcs.resfmt, on),
2527 } :gsub(decimals,rmzeros)
2528 on = update_pdfobjs(attr, streamtext)
2529 os = "<</SMask<</S/Luminosity/G " .. format(pdfetcs.resfmt, on) .. ">>>>"
2530 on, new = update_pdfobjs(os)
2531 local key = add_extgs_resources(on,new)
2532 start_pdf_code()
2533 pdf_literalcode("/%$ gs", key)
2534 if fd_stop then return "standalone" end
2535 return "start"
2536 end
2537

```

Transparency Group

```

2538 pdfetcs.tr_group = { shifts = { } }
2539 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
2540 local function do_preobj_GRP (object, prescript)
2541   local grstate = prescript and prescript.gr_state
2542   if not grstate then return end
2543   local trgroup = pdfetcs.tr_group
2544   if grstate == "start" then
2545     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
2546     trgroup.isolated, trgroup.knockout = false, false
2547     for _,v in ipairs(prescript.gr_type:explode",") do
2548       trgroup[v] = true
2549     end
2550     trgroup.bbox = prescript.mplibgroupbbox:explode":"
2551     put2output[["\beginpgroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
2552   elseif grstate == "stop" then
2553     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
2554     put2output(tableconcat{
2555       "\\egroup",

```

```

2556     format("\\wd\\mplibscratchbox %fbp", urx-llx),
2557     format("\\ht\\mplibscratchbox %fbp", ury-lly),
2558     "\\dp\\mplibscratchbox 0pt",
2559 })
2560 local grattr = format("/Group<</S/Transparency/I %s/K %s>>", trgroup.isolated, trgroup.knockout)
2561 local res = gather_resources()
2562 local bbox = format("%f %f %f %f", llx, lly, urx, ury) :gsub(decimals, rmzeros)
2563 if pdfmode then
2564     put2output(tableconcat{
2565         "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
2566         "/BBox[" .. bbox .. "], grattr, "} resources{" .. res .. "}" .. "\\mplibscratchbox",
2567         "\\luamplibtagasgroupput{" .. trgroup.name .. "}",
2568         [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
2569         [[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]],
2570         [[\box\mplibscratchbox]],
2571         "}\\endgroup",
2572         "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{" ..
2573         "\\setbox\\mplibscratchbox\\hbox{\\hskip", -llx, "bp\\raise", -lly, "bp\\hbox{" ..
2574         "\\useboxresource \\the\\lastsavedboxresourceindex",
2575         "}}\\wd\\mplibscratchbox", urx-llx, "bp\\ht\\mplibscratchbox", ury-lly, "bp",
2576         "\\box\\mplibscratchbox}",
2577     })
2578 else
2579     trgroup.cnt = (trgroup.cnt or 0) + 1
2580     local objname = format("@mplibtrgr%s", trgroup.cnt)
2581     put2output(tableconcat{
2582         "\\special{pdf:boxobj " .. objname .. " bbox " .. bbox .. "}",
2583         "\\unhbox\\mplibscratchbox",
2584         "\\special{pdf:put @resources <<", res, ">>}",
2585         "\\special{pdf:exobj <<", grattr, ">>}",
2586         "\\luamplibtagasgroupput{" .. trgroup.name .. "}",
2587         "\\special{pdf:uxobj " .. objname .. "}",
2588         "}\\endgroup",
2589     })
2590     token.set_macro("luamplib.group." .. trgroup.name, tableconcat{
2591         "\\setbox\\mplibscratchbox\\hbox{\\hskip", -llx, "bp\\raise", -lly, "bp\\hbox{" ..
2592         "\\special{pdf:uxobj " .. objname .. "}",
2593         "}}\\wd\\mplibscratchbox", urx-llx, "bp\\ht\\mplibscratchbox", ury-lly, "bp",
2594         "\\box\\mplibscratchbox",
2595     }, "global")
2596 end
2597 trgroup.shifts[trgroup.name] = { llx, lly }
2598 end
2599 return grstate
2600 end
2601 function luamplib.registergroup (boxid, name, opts)
2602     local box = texgetbox(boxid)
2603     local wd, ht, dp = node.getwhd(box)
2604     local res = (opts.resources or "") .. gather_resources()

```

```

2605 local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
2606 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2607 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2608 if opts.matrix and opts.matrix:find"%a" then
2609     local data = format("mplibtransformmatrix(%s);", opts.matrix)
2610     process(data, "@mplibtransformmatrix")
2611     opts.matrix = format("%f %f %f %f %f %f", tableunpack(luamplib.transformmatrix))
2612 end
2613 local grtype = 3
2614 if opts.bbox then
2615     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2616     grtype = 2
2617 end
2618 if opts.matrix then
2619     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
2620     grtype = opts.bbox and 4 or 1
2621 end
2622 if opts.asgroup then
2623     local t = { isolated = false, knockout = false }
2624     for _, v in ipairs(opts.asgroup:explode, "+") do t[v] = true end
2625     attr[#attr+1] = format("/Group<</S/Transparency/I %s/K %s>>", t.isolated, t.knockout)
2626 end
2627 local trgroup = pdfetcs.tr_group
2628 trgroup.shifts[name] = { get_macro'MPlly', get_macro'MPlly' }
2629 local whd
2630 if pdfmode then
2631     attr = tableconcat(attr) :gsub(decimals, rmzeros)
2632     local index = tex.saveboxresource(boxid, attr, res, true, grtype)
2633     token.set_macro("luamplib.group."..name, tableconcat{
2634         "\\useboxresource ", index,
2635     }, "global")
2636     whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals, rmzeros)
2637 else
2638     trgroup.cnt = (trgroup.cnt or 0) + 1
2639     local objname = format("@mplibtrgr%s", trgroup.cnt)
2640     texsprint {
2641         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2642         "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2643         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2644         "\\special{pdf:bcontent}",
2645         "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
2646         "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2647         "\\special{pdf:put @resources <<, res, ">>}",
2648         "\\special{pdf:exobj <<, tableconcat(attr), ">>}",
2649         "\\special{pdf:econtent}}",
2650     }
2651     token.set_macro("luamplib.group."..name, tableconcat{
2652         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
2653         "\\wd\\mplibscratchbox ", wd, "sp",

```

```

2654     "\\ht\\mplibscratchbox ", ht, "sp",
2655     "\\dp\\mplibscratchbox ", dp, "sp",
2656     "\\box\\mplibscratchbox",
2657   }, "global")
2658   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
2659 end
2660 info("w/h/d of group '%s': %s", name, whd)
2661 end
2662
2663 local function stop_special_effects(fade,opaq,over)
2664   if fade then -- fading
2665     stop_pdf_code()
2666   end
2667   if opaq then -- opacity
2668     pdf_literalcode(opaq)
2669   end
2670   if over then -- color
2671     put2output "\\special{pdf:ec}"
2672   end
2673 end
2674

```

Codes below for inserting PDF literals are mostly from ConTeXt general, with small changes when needed.

```

2675 local function getobjects(result,figure,f)
2676   return figure:objects()
2677 end
2678
2679 function luamplib.convert (result, flusher)
2680   luamplib.flush(result, flusher)
2681   return true -- done
2682 end
2683
2684 local function pdf_textfigure(font,size,text,width,height,depth)
2685   text = text:gsub(".",function(c)
2686     return format("\\hbox{\\char%i}",string.byte(c)) -- kerning happens in metapost : false
2687   end)
2688   put2output("\\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
2689 end
2690
2691 local bend_tolerance = 131/65536
2692
2693 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
2694
2695 local function pen_characteristics(object)
2696   local t = mplib.pen_info(object)
2697   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
2698   divider = sx*sy - rx*ry
2699   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width

```



```

2700 end
2701
2702 local function concat(px, py) -- no tx, ty here
2703   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
2704 end
2705
2706 local function curved(ith,pth)
2707   local d = pth.left_x - ith.right_x
2708   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
2709     d = pth.left_y - ith.right_y
2710     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
2711       return false
2712     end
2713   end
2714   return true
2715 end
2716
2717 local function flushnormalpath(path,open)
2718   local pth, ith
2719   for i=1,#path do
2720     pth = path[i]
2721     if not ith then
2722       pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
2723     elseif curved(ith,pth) then
2724       pdf_literalcode("%f %f %f %f %f %f c",ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
2725     else
2726       pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
2727     end
2728     ith = pth
2729   end
2730   if not open then
2731     local one = path[1]
2732     if curved(pth,one) then
2733       pdf_literalcode("%f %f %f %f %f %f c",pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord)
2734     else
2735       pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2736     end
2737   elseif #path == 1 then -- special case .. draw point
2738     local one = path[1]
2739     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2740   end
2741 end
2742
2743 local function flushconcatpath(path,open)
2744   pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx, ty)
2745   local pth, ith
2746   for i=1,#path do
2747     pth = path[i]
2748     if not ith then

```

```

2749     pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
2750 elseif curved(ith,pth) then
2751     local a, b = concat(ith.right_x,ith.right_y)
2752     local c, d = concat(pth.left_x,pth.left_y)
2753     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
2754 else
2755     pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
2756 end
2757 ith = pth
2758 end
2759 if not open then
2760     local one = path[1]
2761     if curved(pth,one) then
2762         local a, b = concat(pth.right_x,pth.right_y)
2763         local c, d = concat(one.left_x,one.left_y)
2764         pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
2765     else
2766         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2767     end
2768 elseif #path == 1 then -- special case .. draw point
2769     local one = path[1]
2770     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2771 end
2772 end
2773

```

Finally, flush figures by inserting PDF literals.

```

2774 function luamplib.flush (result,flusher)
2775 if result then
2776     local figures = result.fig
2777     if figures then
2778         for f=1, #figures do
2779             info("flushing figure %s",f)
2780             local figure = figures[f]
2781             local objects = getobjects(result,figure,f)
2782             local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
2783             local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2784             local bbox = figure:boundingbox()
2785             local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
2786             if urx < llx then

```

luamplib silently ignores this invalid figure for those that do not contain beginfig ... endfig.
(issue #70) Original code of ConTeXt general was:

```

-- invalid
pdf_startfigure(fignum,0,0,0,0)
pdf_stopfigure()

2787     else

```

For legacy behavior, insert ‘pre-fig’ T_EX code here.

```

2788         if tex_code_pre_mplib[f] then
2789             put2output(tex_code_pre_mplib[f])
2790         end
2791         pdf_startfigure(fignum,llx,lly,urx,ury)
2792         start_pdf_code()
2793         if objects then
2794             local savedpath = nil
2795             local savedhtap = nil
2796             for o=1,#objects do
2797                 local object      = objects[o]
2798                 local objecttype  = object.type

```

The following 10 lines are part of `btex...etex` patch. Again, colors are processed at this stage.

```

2799         local prescript      = object.prescript
2800         prescript = prescript and script2table(prescript) -- prescript is now a table
2801         local cr_over = do_preobj_CR(object,prescript) -- color
2802         local tr_opaq = do_preobj_TR(object,prescript) -- opacity
2803         local fading_ = do_preobj_FADE(object,prescript) -- fading
2804         local trgroup = do_preobj_GRP(object,prescript) -- transparency group
2805         local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
2806         local shading_ = do_preobj_shading(object,prescript) -- shading pattern
2807         if prescript and prescript.mplibtexboxid then
2808             put_tex_boxes(object,prescript)
2809         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
2810             elseif objecttype == "start_clip" then
2811                 local evenodd = not object.istext and object.postscript == "evenodd"
2812                 start_pdf_code()
2813                 flushnormalpath(object.path,false)
2814                 pdf_literalcode(evenodd and "W* n" or "W n")
2815             elseif objecttype == "stop_clip" then
2816                 stop_pdf_code()
2817                 miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2818             elseif objecttype == "special" then

```

Collect \TeX codes that will be executed after flushing. Legacy behavior.

```

2819         if prescript and prescript.postmplibverbtex then
2820             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtex
2821         end
2822         elseif objecttype == "text" then
2823             local ot = object.transform -- 3,4,5,6,1,2
2824             start_pdf_code()
2825             pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
2826             pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
2827             stop_pdf_code()
2828         elseif not trgroup and fading_ ~= "stop" then
2829             local evenodd, collect, both = false, false, false
2830             local postscript = object.postscript
2831             if not object.istext then
2832                 if postscript == "evenodd" then
2833                     evenodd = true

```

```

2834         elseif postscript == "collect" then
2835             collect = true
2836         elseif postscript == "both" then
2837             both = true
2838         elseif postscript == "eoboth" then
2839             evenodd = true
2840             both = true
2841         end
2842     end
2843     if collect then
2844         if not savedpath then
2845             savedpath = { object.path or false }
2846             savedhtap = { object.htap or false }
2847         else
2848             savedpath[#savedpath+1] = object.path or false
2849             savedhtap[#savedhtap+1] = object.htap or false
2850         end
2851     else

```

Removed from ConTeXt general: color stuff.

```

2852         local ml = object.miterlimit
2853         if ml and ml ~= miterlimit then
2854             miterlimit = ml
2855             pdf_literalcode("%f M",ml)
2856         end
2857         local lj = object.linejoin
2858         if lj and lj ~= linejoin then
2859             linejoin = lj
2860             pdf_literalcode("%i j",lj)
2861         end
2862         local lc = object.linecap
2863         if lc and lc ~= linecap then
2864             linecap = lc
2865             pdf_literalcode("%i J",lc)
2866         end
2867         local dl = object.dash
2868         if dl then
2869             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
2870             if d ~= dashed then
2871                 dashed = d
2872                 pdf_literalcode(dashed)
2873             end
2874         elseif dashed then
2875             pdf_literalcode("[ ] 0 d")
2876             dashed = false
2877         end
2878         local path = object.path
2879         local transformed, penwidth = false, 1
2880         local open = path and path[1].left_type and path[#path].right_type

```

```

2881     local pen = object.pen
2882     if pen then
2883         if pen.type == 'elliptical' then
2884             transformed, penwidth = pen_characteristics(object) -- boolean, value
2885             pdf_literalcode("%f w",penwidth)
2886             if objecttype == 'fill' then
2887                 objecttype = 'both'
2888             end
2889         else -- calculated by mplotlib itself
2890             objecttype = 'fill'
2891         end
2892     end

```

Added : shading

```

2893     local shade_no = do_preobj_SH(object,prescript) -- shading
2894     if shade_no then
2895         pdf_literalcode"q /Pattern cs"
2896         objecttype = false
2897     end
2898     if transformed then
2899         start_pdf_code()
2900     end
2901     if path then
2902         if savedpath then
2903             for i=1,#savedpath do
2904                 local path = savedpath[i]
2905                 if transformed then
2906                     flushconcatpath(path,open)
2907                 else
2908                     flushnormalpath(path,open)
2909                 end
2910             end
2911             savedpath = nil
2912         end
2913         if transformed then
2914             flushconcatpath(path,open)
2915         else
2916             flushnormalpath(path,open)
2917         end
2918         if objecttype == "fill" then
2919             pdf_literalcode(evenodd and "h f*" or "h f")
2920         elseif objecttype == "outline" then
2921             if both then
2922                 pdf_literalcode(evenodd and "h B*" or "h B")
2923             else
2924                 pdf_literalcode(open and "S" or "h S")
2925             end
2926         elseif objecttype == "both" then
2927             pdf_literalcode(evenodd and "h B*" or "h B")

```

```

2928         end
2929     end
2930     if transformed then
2931         stop_pdf_code()
2932     end
2933     local path = object.htap

```

How can we generate an htap object? Please let us know if you have succeeded.

```

2934     if path then
2935         if transformed then
2936             start_pdf_code()
2937         end
2938         if savedhtap then
2939             for i=1,#savedhtap do
2940                 local path = savedhtap[i]
2941                 if transformed then
2942                     flushconcatpath(path,open)
2943                 else
2944                     flushnormalpath(path,open)
2945                 end
2946             end
2947             savedhtap = nil
2948             evenodd = true
2949         end
2950         if transformed then
2951             flushconcatpath(path,open)
2952         else
2953             flushnormalpath(path,open)
2954         end
2955         if objecttype == "fill" then
2956             pdf_literalcode(evenodd and "h f*" or "h f")
2957         elseif objecttype == "outline" then
2958             pdf_literalcode(open and "S" or "h S")
2959         elseif objecttype == "both" then
2960             pdf_literalcode(evenodd and "h B*" or "h B")
2961         end
2962         if transformed then
2963             stop_pdf_code()
2964         end
2965     end

```

Added to ConTeXt general: post-object colors and shading stuff. We should beware the q ... Q scope.

```

2966         if shade_no then -- shading
2967             pdf_literalcode("W%s n /MPLibSh%s sh Q",evenodd and "*" or "",shade_no)
2968         end
2969     end
2970 end
2971 if fading_ == "start" then
2972     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}

```

```

2973         elseif trgroup == "start" then
2974             pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
2975         elseif fading_ == "stop" then
2976             local se = pdfetcs.fading.specialeffects
2977             if se then stop_special_effects(se[1], se[2], se[3]) end
2978         elseif trgroup == "stop" then
2979             local se = pdfetcs.tr_group.specialeffects
2980             if se then stop_special_effects(se[1], se[2], se[3]) end
2981         else
2982             stop_special_effects(fading_, tr_opaq, cr_over)
2983         end
2984         if fading_ or trgroup then -- extgs reseted
2985             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2986         end
2987     end
2988 end
2989 stop_pdf_code()
2990 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtex code.

```

2991     for _,v in ipairs(figcontents) do
2992         if type(v) == "table" then
2993             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
2994         else
2995             texsprint(v)
2996         end
2997     end
2998     if #figcontents.post > 0 then texsprint(figcontents.post) end
2999     figcontents = { post = { } }
3000 end
3001 end
3002 end
3003 end
3004 end
3005
3006 function luamplib.colorconverter (cr)
3007     local n = #cr
3008     if n == 4 then
3009         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3010         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3011     elseif n == 3 then
3012         local r, g, b = cr[1], cr[2], cr[3]
3013         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3014     else
3015         local s = cr[1]
3016         return format("%.3f g %.3f G",s,s), "0 g 0 G"
3017     end
3018 end

```

2.2 T_EX package

First we need to load some packages.

```
3019 \ifcsname ProvidesPackage\endcsname
```

We need L^AT_EX 2024-06-01 as we use `ltx.pdf.object_id` when `pdfmanagement` is loaded. But as `fp` package does not accept an option, we do not append the date option.

```
3020 \NeedsTeXFormat{LaTeX2e}
3021 \ProvidesPackage{luamplib}
3022 [2025/12/05 v2.37.7 mplib package for LuaTeX]
3023 \fi
3024 \ifdefined\newluafunction\else
3025 \input ltluatex
3026 \fi
```

In DVI mode, a new XObject (mppattern, mplibgroup) must be encapsulated in an `\hbox`. But this should not affect typesetting. So we use Hook mechanism provided by L^AT_EX kernel. In Plain, `atbegshi.sty` is loaded.

```
3027 \ifnum\outputmode=0
3028 \ifdefined\AddToHookNext
3029 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3030 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3031 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3032 \else
3033 \input atbegshi.sty
3034 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3035 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3036 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3037 \fi
3038 \fi
```

Loading of lua code.

```
3039 \directlua{require("luamplib")}
```

legacy commands. Seems we don't need it, but no harm.

```
3040 \ifx\pdfoutput\undefined
3041 \let\pdfoutput\outputmode
3042 \fi
3043 \ifx\pdfliteral\undefined
3044 \protected\def\pdfliteral{\pdfextension literal}
3045 \fi
```

Set the format for METAPOST.

```
3046 \def\mplibsetformat#1{\directlua{luamplib.setformat("#1")}}
```

`luamplib` works in both PDF and DVI mode, but only DVIPDFMx is supported currently among a number of DVI tools. So we output a info.

```
3047 \ifnum\pdfoutput>0
3048 \let\mplibtoPDF\pdfliteral
3049 \else
```



```

3050 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3051 \ifcsname PackageInfo\endcsname
3052 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3053 \else
3054 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3055 \fi
3056 \fi

    To make mplibcode typeset always in horizontal mode.
3057 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3058 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3059 \mplibnoforcehmode

    Catcode. We want to allow comment sign in mplibcode.
3060 \def\mplibsetupcatcodes{%
3061 %catcode`\{=12 %catcode`\}=12
3062 \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_ =12
3063 \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
3064 }

    Make btex...etex box zero-metric.
3065 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}

    use Transparency Group
3066 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3067 \def\usemplibgroupmain#1{%
3068 \prependtomplibbox\hbox dir TLT\bgroup
3069 \csname luamplib.group.#1\endcsname
3070 \egroup
3071 }
3072 \protected\def\mplibgroup#1{%
3073 \begingroup
3074 \def\MPllx{0}\def\MPlly{0}%
3075 \def\mplibgroupname{#1}%
3076 \mplibgroupgetnexttok
3077 }
3078 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3079 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
3080 \def\mplibgroupbranch{%
3081 \ifx [\nexttok
3082 \expandafter\mplibgroupopts
3083 \else
3084 \ifx\mplibsptoken\nexttok
3085 \expandafter\expandafter\expandafter\mplibgroupskipspace
3086 \else
3087 \let\mplibgroupoptions\empty
3088 \expandafter\expandafter\expandafter\mplibgroupmain
3089 \fi
3090 \fi
3091 }
3092 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}

```

```

3093 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3094 \protected\def\endmplibgroup{\egroup
3095 \directlua{ luamplib.registergroup(
3096 \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3097 )}%
3098 \endgroup
3099 }

```

Patterns

```

3100 {\def\:{\global\let\mplibsptoken= } \: }
3101 \protected\def\mppattern#1{%
3102 \begingroup
3103 \def\mplibpatternname{#1}%
3104 \mplibpatterngetnexttok
3105 }
3106 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3107 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3108 \def\mplibpatternbranch{%
3109 \ifx [\nexttok
3110 \expandafter\mplibpatternopts
3111 \else
3112 \ifx\mplibsptoken\nexttok
3113 \expandafter\expandafter\expandafter\mplibpatternskipsspace
3114 \else
3115 \let\mplibpatternoptions\empty
3116 \expandafter\expandafter\expandafter\mplibpatternmain
3117 \fi
3118 \fi
3119 }
3120 \def\mplibpatternopts[#1]{%
3121 \def\mplibpatternoptions{#1}%
3122 \mplibpatternmain
3123 }
3124 \def\mplibpatternmain{%
3125 \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3126 }
3127 \protected\def\endmppattern{%
3128 \egroup
3129 \directlua{ luamplib.registerpattern(
3130 \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3131 )}%
3132 \endgroup
3133 }

```

simple way to use mplib: \mpfig draw fullcircle scaled 10; \endmpfig

```

3134 \def\mpfiginstancename{@mpfig}
3135 \protected\def\mpfig{%
3136 \begingroup
3137 \futurelet\nexttok\mplibmpfigbranch
3138 }

```

```

3139 \def\mplibmpfigbranch{%
3140   \ifx *\nexttok
3141     \expandafter\mplibprempfig
3142   \else
3143     \ifx [\nexttok
3144       \expandafter\expandafter\expandafter\mplibgobbleoptsmpfig
3145     \else
3146       \expandafter\expandafter\expandafter\mplibmainmpfig
3147     \fi
3148   \fi
3149 }
3150 \def\mplibgobbleoptsmpfig[#1]{\mplibmainmpfig}
3151 \def\mplibmainmpfig{%
3152   \begingroup
3153   \mplibsetupcatcodes
3154   \mplibdomainmpfig
3155 }
3156 \long\def\mplibdomainmpfig#1\endmpfig{%
3157   \endgroup
3158   \directlua{
3159     local legacy = luamplib.legacyverbatim
3160     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3161     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3162     luamplib.legacyverbatim = false
3163     luamplib.everymplib["\mpfiginstancename"] = ""
3164     luamplib.everyendmplib["\mpfiginstancename"] = ""
3165     luamplib.process_mplibcode(
3166       "beginfig(0) "..everympfig.." "..[==[\unexpanded{#1}]==].." "..everyendmpfig.." endfig;",
3167       "\mpfiginstancename")
3168     luamplib.legacyverbatim = legacy
3169     luamplib.everymplib["\mpfiginstancename"] = everympfig
3170     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3171   }%
3172   \endgroup
3173 }
3174 \def\mplibprempfig#1{%
3175   \begingroup
3176   \mplibsetupcatcodes
3177   \mplibdoprempfig
3178 }
3179 \long\def\mplibdoprempfig#1\endmpfig{%
3180   \endgroup
3181   \directlua{
3182     local legacy = luamplib.legacyverbatim
3183     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3184     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3185     luamplib.legacyverbatim = false
3186     luamplib.everymplib["\mpfiginstancename"] = ""
3187     luamplib.everyendmplib["\mpfiginstancename"] = ""

```

```

3188   luaplib.process_mplibcode([===[\unexpanded{#1}]==], "\mpfiginstancename")
3189   luaplib.legacyverbatim = legacy
3190   luaplib.everymplib["\mpfiginstancename"] = everympfig
3191   luaplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3192 }%
3193 \endgroup
3194 }
3195 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3196 \unless\ifcsname ver@luaplib.sty\endcsname
3197 \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3198 \protected\def\mplibcode{%
3199   \begingroup
3200   \futurelet\nexttok\mplibcodebranch
3201 }
3202 \def\mplibcodebranch{%
3203   \ifx \nexttok
3204     \expandafter\mplibcodegetinstancename
3205   \else
3206     \global\let\currentmpinstancename\empty
3207     \expandafter\mplibcodeindeed
3208   \fi
3209 }
3210 \def\mplibcodeindeed{%
3211   \begingroup
3212   \mplibsetupcatcodes
3213   \mplibdocode
3214 }
3215 \long\def\mplibdocode#1\endmplibcode{%
3216   \endgroup
3217   \directlua{luaplib.process_mplibcode([===[\unexpanded{#1}]==], "\currentmpinstancename")}%
3218   \endgroup
3219 }
3220 \protected\def\endmplibcode{endmplibcode}
3221 \else

```

The L^AT_EX-specific part: a new environment.

```

3222 \newenvironment{mplibcode}[1][{}]{%
3223   \xdef\currentmpinstancename{#1}%
3224   \mplibtmptoks{}\ltxdomplibcode
3225 }{}
3226 \def\ltxdomplibcode{%
3227   \begingroup
3228   \mplibsetupcatcodes
3229   \ltxdomplibcodeindeed
3230 }
3231 \def\mplib@mplibcode{mplibcode}
3232 \long\def\ltxdomplibcodeindeed#1\end#2{%
3233   \endgroup

```

```

3234 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3235 \def\mplibtemp@a{#2}%
3236 \ifx\mplib@mplibcode\mplibtemp@a
3237 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]==],"\currentmpinstancename")}%
3238 \end{mplibcode}%
3239 \else
3240 \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3241 \expandafter\ltxdomplibcode
3242 \fi
3243 }
3244 \fi

```

User settings.

```

3245 \def\mplibshowlog#1{\directlua{
3246   local s = string.lower("#1")
3247   if s == "enable" or s == "true" or s == "yes" then
3248     luamplib.showlog = true
3249   else
3250     luamplib.showlog = false
3251   end
3252 }}
3253 \def\mpliblegacybehavior#1{\directlua{
3254   local s = string.lower("#1")
3255   if s == "enable" or s == "true" or s == "yes" then
3256     luamplib.legacyverbatim = true
3257   else
3258     luamplib.legacyverbatim = false
3259   end
3260 }}
3261 \def\mplibverbatim#1{\directlua{
3262   local s = string.lower("#1")
3263   if s == "enable" or s == "true" or s == "yes" then
3264     luamplib.verbatiminput = true
3265   else
3266     luamplib.verbatiminput = false
3267   end
3268 }}
3269 \newtoks\mplibtmptoks

```

\everymplib & \everyendmplib: macros resetting luamplib.every(end)mp lib tables

```

3270 \ifcsname ver@luamplib.sty\endcsname
3271 \protected\def\everymplib{%
3272   \begingroup
3273   \mplibsetupcatcodes
3274   \mplibdoeverymplib
3275 }
3276 \protected\def\everyendmplib{%
3277   \begingroup
3278   \mplibsetupcatcodes
3279   \mplibdoeveryendmplib

```

```

3280 }
3281 \newcommand\mplibdoeverymplib[2][\{%
3282   \endgroup
3283   \directlua{
3284     luampplib.everymplib["#1"] = [===[\unexpanded{#2}]===[
3285   ]%
3286 }
3287 \newcommand\mplibdoeveryendmplib[2][\{%
3288   \endgroup
3289   \directlua{
3290     luampplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===[
3291   ]%
3292 }
3293 \else
3294 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3295 \protected\def\everymplib#1#{%
3296   \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3297   \begingroup
3298   \mplibsetupcatcodes
3299   \mplibdoeverymplib
3300 }
3301 \long\def\mplibdoeverymplib#1{%
3302   \endgroup
3303   \directlua{
3304     luampplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===[
3305   ]%
3306 }
3307 \protected\def\everyendmplib#1#{%
3308   \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3309   \begingroup
3310   \mplibsetupcatcodes
3311   \mplibdoeveryendmplib
3312 }
3313 \long\def\mplibdoeveryendmplib#1{%
3314   \endgroup
3315   \directlua{
3316     luampplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===[
3317   ]%
3318 }
3319 \fi

```

Allow \TeX `dimen`/`color` macros. Now `runscript` does the job, so the following lines are not needed for most cases.

```

3320 \def\mpdim#1{ runscript("luampplibdimen{#1}") }
3321 \def\mpcolor#1#\domplibcolor{#1}}
3322 \def\domplibcolor#1#2{ runscript("luampplibcolor{#1{#2}}") }

```

`mplib`'s number system. Now binary has gone away.

```

3323 \def\mplibnumbersystem#1{\directlua{
3324   local t = "#1"

```

```

3325 if t == "binary" then t = "decimal" end
3326 luamplib.numbersystem = t
3327 }}

```

Settings for .mp cache files.

```

3328 \def\mplibmakenocache#1{\mplibdomakenocache #1,*}
3329 \def\mplibdomakenocache#1,{%
3330   \ifx\empty#1\empty
3331     \expandafter\mplibdomakenocache
3332   \else
3333     \ifx*#1\else
3334       \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
3335       \expandafter\expandafter\expandafter\mplibdomakenocache
3336     \fi
3337   \fi
3338 }
3339 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*}
3340 \def\mplibdocancelnocache#1,{%
3341   \ifx\empty#1\empty
3342     \expandafter\mplibdocancelnocache
3343   \else
3344     \ifx*#1\else
3345       \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
3346       \expandafter\expandafter\expandafter\mplibdocancelnocache
3347     \fi
3348   \fi
3349 }
3350 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

3351 \def\mplibtexttextlabel#1{\directlua{
3352   local s = string.lower("#1")
3353   if s == "enable" or s == "true" or s == "yes" then
3354     luamplib.texttextlabel = true
3355   else
3356     luamplib.texttextlabel = false
3357   end
3358 }}
3359 \def\mplibcodeinherit#1{\directlua{
3360   local s = string.lower("#1")
3361   if s == "enable" or s == "true" or s == "yes" then
3362     luamplib.codeinherit = true
3363   else
3364     luamplib.codeinherit = false
3365   end
3366 }}
3367 \def\mplibglobaltexttext#1{\directlua{
3368   local s = string.lower("#1")
3369   if s == "enable" or s == "true" or s == "yes" then
3370     luamplib.globaltexttext = true

```

```

3371     else
3372         luamplib.globaltexttext = false
3373     end
3374 }}

```

The followings are from ConTeXt general, mostly.
We use a dedicated scratchbox.

```

3375 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

3376 \def\mplibstarttoPDF#1#2#3#4{%
3377   \prependtomplibbox
3378   \hbox dir TLT\bgroup
3379   \xdef\MPllx{#1}\xdef\MPlly{#2}%
3380   \xdef\MPurx{#3}\xdef\MPury{#4}%
3381   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3382   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3383   \parskip0pt%
3384   \leftskip0pt%
3385   \parindent0pt%
3386   \everypar{}%
3387   \setbox\mplibscratchbox\vbox\bgroup
3388   \noindent
3389 }
3390 \def\mplibstoptoPDF{%
3391   \par
3392   \egroup %
3393   \setbox\mplibscratchbox\hbox %
3394     {\hskip-\MPllx bp%
3395      \raise-\MPlly bp%
3396      \box\mplibscratchbox}%
3397   \setbox\mplibscratchbox\vbox to \MPheight
3398     {\vfill
3399      \hsize\MPwidth
3400      \wd\mplibscratchbox0pt%
3401      \ht\mplibscratchbox0pt%
3402      \dp\mplibscratchbox0pt%
3403      \box\mplibscratchbox}%
3404   \wd\mplibscratchbox\MPwidth
3405   \ht\mplibscratchbox\MPheight
3406   \box\mplibscratchbox
3407   \egroup
3408 }

```

Text items have a special handler.

```

3409 \def\mplibtexttext#1#2#3#4#5{%
3410   \begingroup
3411   \setbox\mplibscratchbox\hbox
3412     {\font\temp=#1 at #2bp%
3413      \temp

```



```

3414     #3}%
3415 \setbox\mplibscratchbox\hbox
3416   {\hskip#4 bp%
3417    \raise#5 bp%
3418    \box\mplibscratchbox}%
3419 \wd\mplibscratchbox0pt%
3420 \ht\mplibscratchbox0pt%
3421 \dp\mplibscratchbox0pt%
3422 \box\mplibscratchbox
3423 \endgroup
3424 }

```

Input luamplib.cfg when it exists.

```

3425 \openin0=luamplib.cfg
3426 \ifeof0 \else
3427   \closein0
3428   \input luamplib.cfg
3429 \fi

```

Code for tagpdf

```

3430 \def\luamplibtagtextboxset#1#2{#2}
3431 \let\luamplibnotagtextboxset\luamplibtagtextboxset
3432 \let\luamplibtagasgroupset\relax
3433 \let\luamplibtagasgroupput\luamplibtagtextboxset
3434 \ifcsname SuspendTagging\endcsname\else\endinput\fi
3435 \ifcsname ver@tagpdf.sty\endcsname \else
3436   \ExplSyntaxOn
3437   \keys_define:nn{luamplib/tagging}
3438   {
3439     ,alt          .code:n = { }
3440     ,actualtext   .code:n = { }
3441     ,artifact     .code:n = { }
3442     ,text         .code:n = { }
3443     ,off          .code:n = { }
3444     ,tag          .code:n = { }
3445     ,adjust-BBox  .code:n = { }
3446     ,tagging-setup .code:n = { }
3447     ,instance     .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3448     ,instancename .meta:n = { instance = {#1} }
3449     ,unknown      .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
3450   }
3451 \RenewDocumentCommand\mplibcode{0{}}
3452 {
3453   \tl_gclear:N \currentmpinstancename
3454   \keys_set:ne{luamplib/tagging}{#1}
3455   \mplibtmptoks{}\ltxdomplibcode
3456 }
3457 \cs_set_eq:NN \mplibaltext \use_none:n
3458 \cs_set_eq:NN \mplibactualtext \use_none:n
2025/12/05: \begin{center}\mpfig ... \endmpfig\end{center} raises an Error! as we issue \everypar{}

```

before flushing literals out. It is related to `\partokencontext=2` recently introduced by \TeX . Why we used `vbox` initially? where `hbox` seems to be sufficient. Anyway, among various solutions including `\partokencontext\z@`, `\let\par\@@par`, and `\endgraf`, we here attempt to address the issue by adding the following line, which should have done by \TeX 's `\everypar`.

```

3459 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
3460 \ExplSyntaxOff
3461 \endinput\fi
3462 \ExplSyntaxOn
3463 \tl_new:N \l__luamplib_tag_envname_tl
3464 \tl_new:N \l__luamplib_tag_alt_tl
3465 \tl_new:N \l__luamplib_tag_alt_dflt_tl
3466 \tl_new:N \l__luamplib_tag_actual_tl
3467 \tl_new:N \l__luamplib_tag_struct_tl
3468 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
3469 \bool_new:N \l__luamplib_tag_usetext_bool
3470 \bool_new:N \l__luamplib_tag_bboxcorr_bool
3471 \seq_new:N \l__luamplib_tag_bboxcorr_seq
3472 \tl_new:N \l__luamplib_tag_bboxdraw_tl
3473 \tl_new:N \l__luamplib_BBoxllx_tl
3474 \tl_new:N \l__luamplib_BBoxlly_tl
3475 \tl_new:N \l__luamplib_BBoxurx_tl
3476 \tl_new:N \l__luamplib_BBoxury_tl
3477 \msg_new:nnn {luamplib}{figure-text-reuse}
3478 {
3479   tex-text~box~#1~probably~is~incorrectly~tagged.~
3480   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
3481   Check~the~resulting~PDF.
3482 }
3483 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
3484 {
3485   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
3486   Using~mplibgroup~with~text~mode~is~not~recommended.~
3487   Check~the~resulting~PDF.
3488 }
3489 \msg_new:nnn {luamplib}{alt-text-missing}
3490 {
3491   Alternate~text~for~#1~is~missing.~
3492   Using~the~default~value~'#2'~instead.
3493 }
```

Sockets for tex-text boxes.

```

3494 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
3495 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
3496 \socket_new_plugin:nnn{tagsupport/luamplib/texttext/set}{default}
3497 {
```

TODO: we check text mode here. If we tag text boxes for all modes, we will get a lot of structure-has-no-parent warning; no good-looking, though it seems to be no harm.

```

3498 \bool_if:NTF \l__luamplib_tag_usetext_bool
```

```

3499 {
3500   \tag_mc_end_push:
3501   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
3502   \cs_gset_nopar:cpe {luamplib.taggedbox.#1} {\tag_get:n{struct_num}}

```

TODO: We force an MC. Otherwise a and b in `btex a x b etex` are not tagged.

```

3503   \tag_mc_begin:n{tag=text}
3504   #2
3505   \tag_mc_end:
3506   \tag_struct_end:
3507   \tag_mc_begin_pop:n{}
3508 }
3509 {
3510   \tag_suspend:n{\luamplibtagtextboxset}
3511   #2
3512   \tag_resume:n{\luamplibtagtextboxset}
3513 }
3514 }
3515 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
3516 {
3517   \bool_lazy_and:nnTF
3518   { \l__luamplib_tag_usetext_bool }
3519   { \cs_if_free_p:c {luamplib.notaggedbox.#1} }
3520   {
3521     \tag_resume:n{\mplibputtextbox}
3522     \tag_mc_end:
3523     \cs_if_exist:cTF {luamplib.taggedbox.#1}
3524     {
3525       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
3526       #2
3527       \cs_undefine:c {luamplib.taggedbox.#1}
3528     }
3529     {
3530       \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
3531       \tag_mc_begin:n{}
3532       \int_set:Nn \l_tmpa_int {#1}
3533       \tag_mc_reset_box:N \l_tmpa_int
3534       #2
3535       \tag_mc_end:
3536     }
3537     \tag_mc_begin:n{artifact}
3538   }
3539   {
3540     \int_set:Nn \l_tmpa_int {#1}
3541     \tag_mc_reset_box:N \l_tmpa_int
3542     #2
3543   }
3544 }
3545 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}

```

```

3546 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
3547 \cs_set_nopar:Npn \luamplibtagtextboxset
3548 {
3549   \tag_socket_use:nnn{luamplib/texttext/set}
3550 }

```

For tex-text boxes starting with [taggingoff], which we will not tag at all. They will be just in the artifact MC-chunks.

```

3551 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
3552 {
3553   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
3554   \bool_set_false:N \l__luamplib_tag_usetext_bool
3555   \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
3556   \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
3557   \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
3558 }
3559 \cs_set_nopar:Npn \mplibputtextbox #1
3560 {
3561   \vbox to 0pt{\vss\hbox to 0pt{
3562     \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}
3563     \hss}}
3564 }

```

TODO: Not sure whether asgroup/mplibgroup with text mode will be tagged correctly. Probably not. At least, this will raise a warning.

```

3565 \cs_set_nopar:Npn \luamplibtagasgroupset
3566 {
3567   \bool_set_false:N \l__luamplib_tag_usetext_bool
3568 }
3569 \cs_set_nopar:Npn \luamplibtagasgroupput
3570 {
3571   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
3572   \tag_socket_use:nnn{luamplib/mplibgroup/put}
3573 }

```

A socket for mplibgroup. Again, we issue a warning upon text mode.

```

3574 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
3575 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
3576 {
3577   \cs_if_free:cT {luamplib.mplibgroup.text.#1}
3578   {
3579     \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
3580     \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
3581   }
3582   \tag_mc_end:
3583   \tag_mc_begin:n{tag=text}
3584   #2
3585   \tag_mc_end:
3586   \tag_mc_begin:n{artifact}
3587 }

```

```
3588 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}
```

A macro for BBox attribute

```
3589 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
3590 {
3591   \tl_set:Nx \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
3592   \tex_savepos:D
3593   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
3594   \tl_set:Nx \l__luamplib_BBox_llx_tl
3595     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
3596   \tl_set:Nx \l__luamplib_BBox_lly_tl
3597     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
3598   \tl_set:Nx \l__luamplib_BBox_urx_tl
3599     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
3600   \tl_set:Nx \l__luamplib_BBox_ury_tl
3601     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
3602   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
3603   {
3604     \int_zero:N \l_tmpa_int
3605     \tl_map_inline:nn
3606     {
3607       \l__luamplib_BBox_llx_tl
3608       \l__luamplib_BBox_lly_tl
3609       \l__luamplib_BBox_urx_tl
3610       \l__luamplib_BBox_ury_tl
3611     }
3612     {
3613       \int_incr:N \l_tmpa_int
3614       \tl_set:Nx ##1
3615       {
3616         \fp_eval:n
3617         {
3618           ##1
3619           +
3620           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
3621         }
3622       }
3623     }
3624   }
3625   \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
3626   {
3627     /O /Layout /BBox [
3628       \l__luamplib_BBox_llx_tl\c_space_tl
3629       \l__luamplib_BBox_lly_tl\c_space_tl
3630       \l__luamplib_BBox_urx_tl\c_space_tl
3631       \l__luamplib_BBox_ury_tl
3632     ]
3633   }
3634   \bool_if:NT \l__tag_graphic_debug_bool
```

```

3635 {
3636   \iow_log:e
3637   {
3638     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
3639     \l__luamplib_BBox_llx_tl\c_space_tl
3640     \l__luamplib_BBox_lly_tl\c_space_tl
3641     \l__luamplib_BBox_urx_tl\c_space_tl
3642     \l__luamplib_BBox_ury_tl
3643   }
3644   \sys_if_output_pdf:TF
3645   {
3646     \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
3647     {
3648       \pdfextension save\relax
3649       \opacity_select:n{0.5} \color_select:n{red}
3650       \pdfextension literal~text
3651       {
3652         \l__luamplib_BBox_llx_tl\c_space_tl
3653         \l__luamplib_BBox_lly_tl\c_space_tl
3654         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3655         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3656         re~f
3657       }
3658       \pdfextension restore\relax
3659     }
3660   }
3661   {
3662     \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
3663     {
3664       \special{pdf:bcontent}
3665       \opacity_select:n{0.5} \color_select:n{red}
3666       \special{pdf:code~
3667         1~0~0~1~
3668         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
3669         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
3670         cm
3671       }
3672       \special{pdf:code~
3673         \l__luamplib_BBox_llx_tl\c_space_tl
3674         \l__luamplib_BBox_lly_tl\c_space_tl
3675         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3676         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3677         re~f
3678       }
3679       \special{pdf:econtent}
3680     }
3681   }
3682 }
3683 }

```

Sockets for main process

```

3684 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
3685 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
3686 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
3687 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
3688 {
3689   \tag_mc_end_push:
3690   \tl_if_empty:NT\l__luamplib_tag_alt_tl
3691   {
3692     \tl_if_empty:eTF{#1}
3693     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
3694     { \tl_set:Ne \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
3695     \msg_warning:nnVV{luamplib}{alt-text-missing}
3696     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
3697   }
3698   \tag_struct_begin:n
3699   {
3700     tag=\l__luamplib_tag_struct_tl,
3701     alt=\l__luamplib_tag_alt_tl,
3702   }
3703   \tag_mc_begin:n{}
3704 }
3705 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
3706 {
3707   \__luamplib_tag_bbox_attribute:n {#1}
3708   #2
3709   \tl_use:N \l__luamplib_tag_bbox_draw_tl
3710   \tag_mc_end:
3711   \tag_struct_end:
3712   \tag_mc_begin_pop:n{}
3713 }
3714 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
3715 {
3716   \tag_mc_end_push:
3717   \tag_struct_begin:n
3718   {
3719     tag=Span,
3720     actualtext=\l__luamplib_tag_actual_tl,
3721   }
3722   \tag_mc_begin:n{}
3723 }
3724 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
3725 {
3726   #2
3727   \tag_mc_end:
3728   \tag_struct_end:
3729   \tag_mc_begin_pop:n{}
3730 }
3731 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}

```

```

3732 {
3733   \tag_mc_end_push:
3734   \tag_mc_begin:n{artifact}
3735 }
3736 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
3737 {
3738   #2
3739   \tag_mc_end:
3740   \tag_mc_begin_pop:n{ }
3741 }

```

A socket for tagging init, so that we can declare `\SetKeys[luamplib/tagging]{...}` anywhere in the document.

```

3742 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
3743 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
3744 {
3745   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
3746   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
3747 }
3748 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
3749 {
3750   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
3751   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

In vmode, hmode will be forced by `\noindent` upon actualtext and text modes.

```

3752 \prependtomplibbox \mplibnoforcehmode
3753 \mode_if_vertical:T { \noindent \aftergroup\par }
3754 }
3755 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
3756 {
3757   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
3758   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
3759 }
3760 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
3761 {
3762   \bool_set_true:N \l__luamplib_tag_usetext_bool
3763   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
3764   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
3765   \prependtomplibbox \mplibnoforcehmode
3766   \mode_if_vertical:T { \noindent \aftergroup\par }
3767 }
3768 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
3769 {
3770   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
3771   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
3772 }
3773 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```

3774 \keys_define:nn{luamplib/tagging}

```



```

3775 {
3776   ,alt .code:n =
3777   {
3778     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
3779     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
3780   }
3781   ,actualtext .code:n =
3782   {
3783     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
3784     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
3785   }
3786   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
3787   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
3788   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
3789   ,tag .code:n =
3790   {
3791     \str_case:nnF {#1}
3792     {
3793       {false} { \keys_set:nn {luamplib/tagging} {off} }
3794       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
3795     }
3796     {
3797       \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
3798       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
3799     }
3800   }
3801   ,adjust-BBox .code:n =
3802   {
3803     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
3804     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
3805   }
3806   ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
3807 }
3808 \keys_define:nn {luamplib/instance}
3809 {
3810   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3811   ,instancename .meta:n = { instance = {#1} }
3812   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
3813 }

```

Redefine our macros

```

3814 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
3815 {
3816   \prependtomplibbox
3817   \hbox dir~TLT\bgroup
3818     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
3819     \xdef\MPllx{#1}\xdef\MPlly{#2}%
3820     \xdef\MPurx{#3}\xdef\MPury{#4}%
3821     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%

```

```

3822 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3823 \parskip0pt
3824 \leftskip0pt
3825 \parindent0pt
3826 \everypar{}%
3827 \setbox\mplibscratchbox\vbox\bgroup
3828 \tag_suspend:n{\mplibstarttoPDF}
3829 \noindent
3830 }
3831 \cs_set_nopar:Npn \mplibstoptoPDF
3832 {
3833 \par
3834 \egroup
3835 \setbox\mplibscratchbox\hbox
3836 {\hskip-\MPllx bp
3837 \raise-\MPlly bp
3838 \box\mplibscratchbox}%
3839 \setbox\mplibscratchbox\vbox to \MPheight
3840 {\vfill
3841 \hsize\MPwidth
3842 \wd\mplibscratchbox0pt
3843 \ht\mplibscratchbox0pt
3844 \dp\mplibscratchbox0pt
3845 \box\mplibscratchbox}%
3846 \wd\mplibscratchbox\MPwidth
3847 \ht\mplibscratchbox\MPheight
3848 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
3849 \egroup
3850 }
3851 \RenewDocumentCommand\mplibcode{0{}}
3852 {
3853 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
3854 \tl_gclear:N \currentmpinstancename
3855 \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
3856 \keys_set:nV {luamplib/instance} \l_tmpa_tl
3857 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
3858 \tag_socket_use:n{luamplib/figure/init}
3859 \mplibtmptoks{}\ltxdomplibcode
3860 }
3861 \RenewDocumentCommand\mpfig{s 0{}}
3862 {
3863 \begingroup
3864 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
3865 \keys_set_known:ne {luamplib/tagging} {#2}
3866 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
3867 \tag_socket_use:n{luamplib/figure/init}
3868 \IfBooleanTF{#1} { \mplibprempfig * }
3869 { \mplibmainmpfig }
3870 }

```

```

3871 \RenewDocumentCommand\usemplibgroup{0{} m}
3872 {
3873   \begingroup
3874   \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
3875   \keys_set_known:ne {luamplib/tagging} {#1}
3876   \tag_socket_use:n{luamplib/figure/init}
3877   \prependtomplibbox\hbox dir~TLT\bgroup
3878     \tag_socket_use:nn{luamplib/figure/begin}{#2}
3879     \setbox\mplibscratchbox\hbox\bgroup
3880       \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
3881       \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
3882       \egroup
3883       \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
3884     \egroup
3885   \endgroup
3886 }

```

Allow setting alt/actual text within METAPOST code. Of course we can use them in \TeX code as well.

```

3887 \cs_new_nopar:Npn \mplibalttext #1
3888 {
3889   \tl_set:Ne \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
3890 }
3891 \cs_new_nopar:Npn \mplibactualtext #1
3892 {
3893   \tl_set:Ne \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
3894 }
3895 \ExplSyntaxOff

```

That's all folks!

3 The GNU GPL License v2

The GPL requires the complete license text to be distributed along with the code. I recommend the canonical source, instead: <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>. But if you insist on an included copy, here it is. You might want to zoom in.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it. For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software. Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

- This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".
- Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
- You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
- You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
- You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
 - If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when

you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

- You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or
 - Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

- You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
- You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
- Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
- If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

- If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
- The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

- If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

- BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
- IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) yyyy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program
"Gnomovision" (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subcomponent library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.