

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/04/28 v2.41.0

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>	6
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>	8
1.1.15	About cache files	8
1.1.16	About figure box metric	9
1.1.17	<code>luamplib.cfg</code>	9
1.1.18	Tagged PDF	9
1.2	METAPOST	11
1.2.1	<code>mplibdimen, mplibcolor</code>	11
1.2.2	<code>mplibtexcolor, mplibrgbtexcolor</code>	11
1.2.3	<code>withmplibcolors</code>	11
1.2.4	<code>withtransparency</code>	12

1.2.5	<code>withmplibopacities</code>	12
1.2.6	<code>withshadingmethod</code>	13
1.2.7	<code>withfademethod</code>	15
1.2.8	<code>mplibgraphicstext</code>	15
1.2.9	<code>mplibglyph</code>	16
1.2.10	<code>mplibdrawglyph</code> , and its friends	17
1.2.11	<code>mpliboutlinetext</code>	17
1.2.12	<code>\mppattern</code> , <code>withmppattern</code>	18
1.2.13	<code>asgroup</code>	20
1.2.14	<code>\mplibgroup</code>	23
1.2.15	<code>withmaskinggroup</code>	24
1.2.16	<code>mpliblength</code> , <code>mplibuclength</code>	25
1.2.17	<code>mplibsubstring</code> , <code>mplibucsubstring</code>	25
1.3	<code>Lua</code>	25
1.3.1	<code>runscript</code>	25
1.3.2	<code>luamplib.instances</code>	26
1.3.3	<code>luamplib.process_mplibcode</code>	26
1.3.4	<code>luamplib.registerpattern</code>	27
1.3.5	<code>luamplib.registergroup</code>	27
2	Implementation	28
2.1	<code>Lua module</code>	28
2.2	<code>TeXpackage</code>	98
3	The GNU GPL License v2	119

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` to run a \TeX code. `VerbatimTeX` $\langle string \rangle$ is a more versatile macro corresponding to `verbatimtex` command. Of course the behavior cannot be the same as the stand-alone `mpost`, so that 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 v2.7 it works in DVI mode as well, though DVI_{PDF}Mx 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: \TeX , METAPOST, and Lua interfaces.

1.1 \TeX

1.1.1 `\mplibforcehmode`

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

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

`\everymplib` and `\everyendmplib` redefine the Lua table entry 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` $\langle format name \rangle$.

N.B. As *metafun* is such a complicated format, we cannot support all the 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 § 1.2.11). You can try other effects as well, though we did not fully tested their proper functioning.

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

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=⟨numeric⟩"` to the sentence. ($0 \leq \langle \text{numeric} \rangle \leq 1$)

From v2.36, `withtransparency` is available with *plain* format as well. See below § 1.2.4.

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 T_EX 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 § 1.2.6.

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 $\langle \text{string} \rangle$ should be "" (empty), "isolated", "knockout", or "isolated,knockout". Beware that currently many of the PDF rendering applications, except Adobe Acrobat and Foxit Editor, cannot properly render the isolated or knockout effect.

Transparency group is available with *plain* format as well with extended functionality. See below § 1.2.13.

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 T_EX side interface for `luamplib.showlog`.

1.1.6 `\mpliblegacybehavior{enable|disable}`

Legacy behavior By default, `\mpliblegacybehavior{enable}` is already declared for backward compatibility, in which case T_EX 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;
```

²As for user's setting, `enable`, `true` and `yes` are identical; all others are identical to `disable`.

³But the recommended way to reuse a figure is using `\mplibgroup` command. See below § 1.2.14.

```

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

```


N.B. `\endgraf` should be used instead of `\par` inside `mplibcode` environment.

On the other hand, \TeX code in `verbatimtex ... etex` between `beginfig()` and `endfig` will be inserted after flushing out the `METAPOST` figure. An example:⁴

```

\mplibcode
D := sqrt(2)**9;
beginfig(0);
  draw fullcircle scaled D;
  VerbatimTeX("\gdef\Dia{" & decimal D & "}");
endfig;
\endmplibcode
diameter: \Dia bp.

```



New and recommended way By contrast, when `\mpliblegacybehavior{disable}` is declared, any `verbatimtex ... etex`, along with `btex ... etex`, will be run sequentially one by one. So, some \TeX code in `verbatimtex ... etex` will have effect on `btex ... etex` codes thereafter.

```

\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}

```

ABC **DEF GHI**

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 .

⁴But the recommended way to access `METAPOST` variables from \TeX (or Lua) side is to use Lua code via `luamplib`.instances. For details see below § 1.3.2.

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 other hand, `\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. The 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 \LaTeX environment `mplibcode` or Plain \TeX commands `\mplibcode ... \endmplibcode`. The syntax for \LaTeX 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 the 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.

```
draw origin--(.4\mpdim{\linewidth},0)
  withpen pencircle scaled 4 dashed evenly scaled 4
  withcolor \mpcolor{orange} ;
```



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` command, in principle). See the example above at § 1.1.12. The optional `[...]` denotes the option of `xcolor`'s `\color` command. For spot colors, `l3color` module is well supported in PDF and DVI mode. Package `colorspace` is supported as well in PDF mode, but could conflict with `luamplib`'s special features such as shading when `\DocumentMetadata`, ie. PDF management code, is not loaded.

N.B. Formerly, only the first object would have been colored as intended among multiple graphical objects in a `METAPOST` image, because `\mpcolor` always produced `withprescript` command internally. Since v2.38.1, now that `\mpcolor` returns a `METAPOST` color expression if possible, users can issue the sentence as follows without worrying about the location of the color command:

```
draw image (drawarrow (left--right) scaled 5)
  scaled 8
  withcolor \mpcolor{red!50} ;
```



N.B. Be aware, however, that even after v2.38.1 `\mpcolor` still inserts `withprescript` command when the color specified is a spot color (or named color in DVI mode). Users therefore have to revise the code so that the color can have effect inside the image. For instance:

```
draw image (drawarrow (left--right) scaled 5)
  scaled 8
  withcolor \mpcolor{spotA}
  withoutcolor ;
```

or preferably,

```
draw image (drawarrow (left--right) scaled 5 withcolor \mpcolor{spotA})
scaled 8 ;
```

1.1.14 `\mpfig ... \endmpfig`

Besides the `mplibcode` environment (for \LaTeX) 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 1/3[red,white]); }
\mpfig* input boxes \endmpfig
\mpfig
  circleit.a(btex Box 1 etex); drawboxed(a);
\endmpfig
```

Box 1

Users can change the instance name (default value: `@mpfig`) 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` is not true.

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 $\langle filename \rangle$ is a filename without .mp extension. Note that .mp files under \$TEXMFMAIN/metapost/base and \$TEXMFMAIN/metapost/context/base are already registered by default.

N.B. `\mplibmakenocache{*}` will suppress making cache files. Use it at your own risk.

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{text}"`;

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.⁵

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{text}"`;

⁵It 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.

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 T_EX-text boxes (such as `btex ... etex`, excluding pictures made by `infont` operator). If in vertical mode, horizontal mode will be forced by `\noindent` command.⁶ BBox info will not be added. This key is intended for figures the meaning of which is the sequence of texts in the T_EX-text boxes in the order they are drawn in the figure.

N.B. Within text-mode figures, reusing T_EX-text boxes is strongly discouraged.

Note that the text in a T_EX-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 12down ;
    draw TEX "[taggingoff]  $\sqrt{5}$ " shifted 24down ;
    draw maketext " $\sqrt{7}$ " shifted 36down ;
    draw mplibgraphictext " $\sqrt{x}$ " shifted 48down ;
  endfig;
\end{mplibcode}
```

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

tag=*<name>* You can choose a tag name, default value being `Figure`.⁷ For instance, you can set `tag=Formula, alt=<text>` to get a `Formula` element with its alternate text.⁸

adjust-BBox=*<dimens>* 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=*<key-val list>* This key accepts as its value the list of key-value options mentioned so far.

You can set these options anywhere in the document by declaring `\SetKeys[luamplib/tagging]{<key-val list>}`, which will affect `mplib` figures thereafter in the scope. And the options listed above are provided for `\mpfig` and `\usemplibgroup` (see [below § 1.2.13](#)) commands as well.

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

⁶The key `text` also shares the limitation mentioned in the previous footnote.

⁷The option `tag=false`, however, is a synonym of the `off` key.

⁸Beware that this bypasses L^AT_EX's regular math formula tagging, for which the `text` key is needed.

```

\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=<name>` or `instancename=<name>` is also allowed in addition to the raw instance name as shown above.

1.2 METAPOST

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

`mplibdimen <string>` and `mplibcolor <string>` 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`.

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

`mplibtexcolor <string>` is a METAPOST operator that converts a \TeX color expression to a METAPOST color expression, that can be used anywhere color expression is expected as well as after the `withcolor` command.⁹ For instance:

```

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

```

But the result may vary in its color model (gray/rgb/cmyk) according to the given \TeX color. 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.

N.B. Spot colors are forced to cmyk or rgb model, so these operators are not recommended for spot colors.

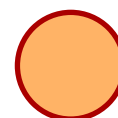
1.2.3 `withmplibcolors (...)`

Unlike the `withcolor` command, users can specify one color for filling and another color for stroking using the macro `withmplibcolors` at the end of a sentence. The syntax is `withmplibcolors`

⁹Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color or a named color in DVI mode.

(*fill color expr*), (*stroke color expr*)). When the argument is in string type, it is regarded as the color expression of \TeX side. A simple example (see also the example at § 1.2.10):

```
filldraw fullcircle scaled 40
  withpen pencircle scaled 2
  withhplibcolors ("orange!60", 2/3red) ;
```

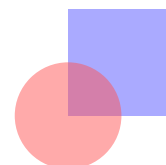


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 withtransparency (... , ...)

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

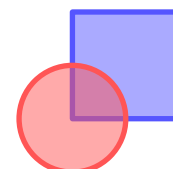
```
\mpfig
fill unitsquare scaled 40
  withcolor 1/3[blue,white]
  withtransparency (1, 0.5)      % or ("normal", 0.5)
;
fill fullcircle scaled 40
  withcolor 1/3[red,white]
  withtransparency (1, 0.5)
;
\endmpfig
```



1.2.5 withhplibopacities (... , ... , ...)

By analogy with the macro *withhplibcolors* (see above § 1.2.3), the macro *withhplibopacities* is also provided. The syntax is *withhplibopacities* (*number* | *string*), (*numeric*), (*numeric*). The first argument is the same as that of *withtransparency* command described above at § 1.2.4; the latter two arguments are numeric expressions denoting *fill opacity* and *stroke opacity* respectively. It is more efficient than issuing two sentences with different opacities.

```
\mpfig
pickup pencircle scaled 2;
filldraw unitsquare scaled 40
  withcolor 1/3[blue,white]
  withhplibopacities (1, 1/2, 1)      % or ("normal", 1/2, 1)
;
filldraw fullcircle scaled 40
  withcolor 1/3[red,white]
  withhplibopacities (1, 1/2, 1)
;
\endmpfig
```



1.2.6 ... 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* as well as paths can have shading effect. The term *textual picture* here means a picture generated by *btex* ... *etex*, *texttext*, *TEX*, *maketext*, *mplibgraphictext* (see below § 1.2.8), or *infont* operator, though technically only the last one is a true textual picture. Note that the picture, including transparency group, in which the objects are filled *without* color can also be regarded as a textual picture (e.g., see below § 1.2.10, particularly the first *example* of tiling pattern at § 1.2.12; see also § 1.2.13 and § 1.2.14).

```
draw btex \bfseries\TeX etex rotated 15 scaled 6
  withshadingmethod "linear"
  withshadingvector (0,3)
  withshadingstep (
    withshadingfraction 1/2
    withshadingcolors (red,green)
  )
  withshadingstep (
    withshadingfraction 1
    withshadingcolors (green,blue)
  ) ;
```



- When shading a picture generated by 'infont' operator or that has multiple components, the effect of *withshadingvector* and that of *withshadingdirection* will be the same, as *luamplib* considers only the bounding box of the picture.
- In addition to those mimicking *metafun*'s, a few more optional macros are available: *withshadingpoints*, *withshadingcenters*, *withshadingextend*, and *withshadingstroke*.

The syntax is *<path>* | *<textual picture>* *withshadingmethod* *<string>*, where the latter shall be either "linear" or "circular". The balance of this subsection is to explain additional optional macros. Above all, there are two ways in specifying the shading coordinates, of which you can choose the more convenient one. First, the way that mimicks the *metafun*'s:

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 being (0,2).

withshadingorigin *<pair>* The center of both starting and ending circles, default value being center p, where p is the operand of *withshadingmethod*.

withshadingcenter *<pair>* Value to specify the 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.

withshadingradius $\langle pair \rangle$ Radii of starting and ending circles. This is no-op in linear mode.
Default value: $(0, \text{abs}(\text{center } p - \text{urcorner } p))$

withshadingfactor $\langle numeric \rangle$ Multiplier of the radii, default value being 1.2. This is no-op in linear mode.

withshadingtransform $\langle string \rangle$ where $\langle string \rangle$ shall be "yes" (respect transform) or "no" (ignore transform). Default value: "no" for pictures made by infont operator or having multiple components; "yes" for all other cases.

Secondly, the way provided by luamplib only:

withshadingpoints $(\langle pair \rangle, \langle pair \rangle)$ In linear mode, values to specify directly the starting and ending points: you can use it instead of withshadingvector or withshadingdirection. In circular mode, the centers of starting and ending circles: it could be easier than issuing two macros withshadingorigin and withshadingcenter. Note that, within the macro, both **withshadingfactor** 1 and **withshadingtransform** "no" are already declared.

withshadingcenters $(\langle pair \rangle, \langle pair \rangle)$ Synonym of withshadingpoints. Normally accompanied by withshadingradius which has the same meaning as described above.

Now, optional macros common to the both ways:

withshadingstep $(...)$ For combined shading of more than two colors.

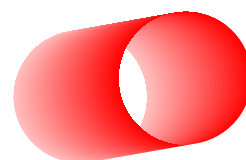
withshadingfraction $\langle numeric \rangle$ Fractional number of each shading step, and so only meaningful within withshadingstep.

withshadingcolors $(\langle color \text{ expr} \rangle, \langle color \text{ expr} \rangle)$ Starting and ending colors, default value being (white, black). String-type argument is regarded as the color expression of T_EX side.

withshadingdomain $\langle pair \rangle$ Limiting values of parametric variable that varies on the axis of color gradient, default value being $(0, 1)$. Of course the values can be negative or greater than 1.

withshadingextend $(\langle boolean \rangle, \langle boolean \rangle)$ Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

```
\mpfig
  path p[];
  p1 = fullcircle scaled 50;
  p2 = fullcircle scaled 50 shifted 40 right;
  fill (subpath (2,6) of p1 -- subpath (-2,2) of p2 -- cycle) rotated 10
    withshadingmethod "circular"
    withshadingcenters (center p1, center p2 rotated 10)
    withshadingradius (25, 25)
    withshadingcolors (3/4[red,white], red)
    withshadingextend (false, false) ;
\endmpfig
```



withshadingstroke $\langle string \rangle$ where $\langle string \rangle$ shall be "yes" or "no". Only meaningful when the shading object is a $\langle path \rangle$; if "yes", we get the path stroked and *then* shaded. It is more efficient than issuing two sentences.

1.2.7 ... withfademethod ...

This is a METAPOST command which makes the color of an object gradiently transparent, a.k.a. *fading*. 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 object of fading can be a picture as well as a path; (2) you cannot make gradient colors, but can only make gradient opacity. Technically speaking, this command generates and applies a special kind of masking transparency group described below at § 1.2.15.

Related macros to control optional values are:

withfadeopacity ($\langle numeric \rangle$, $\langle numeric \rangle$) sets the starting opacity and the ending opacity, default value being (1, 0). '1' denotes full color; '0' full transparency.

withfadevector ($\langle pair \rangle$, $\langle pair \rangle$) 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 ($\langle numeric \rangle$, $\langle numeric \rangle$) 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 ($\langle pair \rangle$, $\langle pair \rangle$) 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](#) at § 1.2.13 on the analogous macro **withgroupbbox**.

An example:

```
draw
  btex \includegraphics[width=100bp]{mill} etex
  withfademethod "circular"
  withfaderadius (20, 50)
  withfadeopacity (1, 0) ;
```



1.2.8 mplibgraphicstext ...

mplibgraphicstext $\langle string \rangle$ is a METAPOST operator, the effect of which is similar to that of Con-TeXt's **graphicstext** or our own **mpliboutlinetext** (see below § 1.2.11). However the syntax is somewhat different.

```
draw mplibgraphicstext "$\sqrt{2+x}$"
```

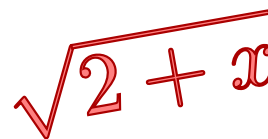


```

rotated 10 scaled 3
fakebold 2.5
fillcolor "red!50"
drawcolor 2/3 red
;

```

% fontspec option
% color expression
% or strokecolor 2/3 red



fakebold, fillcolor and drawcolor (or strokecolor) are optional; default values are 2, "white" and "black" respectively.¹⁰ When the color expression is given in string type, it is regarded as color, xcolor or l3color's expression. All from mpilibgraphicstext to the end of sentence will compose an anonymous picture, which can be drawn or assigned to a variable. Incidentally, withfillcolor and withdrawcolor are synonyms of fillcolor and drawcolor, hopefully to be compatible with graphicstext.

N.B. In some cases, especially when processing complicated T_EX code, mpilibgraphicstext will produce better results than ConT_EXt or even than our own mpiliboutlinetext, not to mention the much smaller PDF file size. There are, however, some limitations such that you can't apply shading (gradient colors) to the text with *metafun*'s withshademethod.¹¹ Again, in DVI mode, unicode-math package is needed for math formulae, as we cannot embolden type1 fonts in DVI mode. But the most critical limitation is that, unlike mpiliboutlinetext, you cannot manipulate the shape of outline paths, because the returned picture is basically a btex ... etex picture.

1.2.9 mpilibglyph ... of ...

METAPOST operator `mpilibglyph <number> | <string> of <number> | <string>` returns a METAPOST picture containing outline paths of a glyph in OpenType, TrueType or Type1 (.pfb) fonts. When a TFM font is specified, METAPOST primitive glyph will be called.

```

mpilibglyph 50 of \fontid\font          % slot 50 of current font
mpilibglyph "Q" of "TU/TeXGyrePagella(0)/m/n/10" % font csname
mpilibglyph "Q" of "texgyrepagella-regular.otf" % raw filename
mpilibglyph "R" of "utmr8a.pfb" % raw filename (type1 font)
mpilibglyph "Q" of "Times.ttc(2)" % subfont number
mpilibglyph "Q" of "SourceHanSansK-VF.otf[Regular]" % instance name
mpilibglyph "R" of "SourceHanSansK-VF.otf[wght=800]" % axis names & values

```

Both arguments before and after '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 T_EX 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, or names and values for axis feature, of a variable font.

N.B. Regrettably we have some bug in processing not a few glyphs in cmr10.pfb and its family (or maybe other) Type1 fonts.¹² If that happens, consider using glyph operator instead of mpilibglyph.

¹⁰Users can use the withmpilibcolors macro instead of fillcolor and drawcolor options. See § 1.2.3 on this macro.

¹¹But this limitation is now lifted by the introduction of withshadingmethod. See above § 1.2.6.

¹²The bug seems to be fixed in font-cff.lmt contained in ConT_EXt mkx1, but current luaotfload is based on font-

1.2.10 `mplibdrawglyph ...`, `mplibstrokeglyph ...`, `mplibfillandstrokeglyph ...`

As the structure of the picture returned by `mplibglyph` is quite similar to the result of `glyph` primitive, METAPOST's `draw` command will fill the inner path of the picture with the background color. In contrast, `mplibdrawglyph` *<picture>* 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.

N.B. 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 additionally declare `withpostscript "evenodd"` to the last path.

N.B. By the way, when you want fill-and-stroke effect, issuing `filldraw` command to the last path will not always produce what you want: in such cases, you have to issue the command `draw` *<the last path>* `withpostscript "both"` (or `"eoboth"` to apply even-odd rule).¹³

As this could be somewhat annoying to users, `luamplib` v2.38.0 or later provides the following commands as well: `mplibfillandstrokeglyph` *<picture>*, `mplibstrokeglyph` *<picture>*, and `mplibfillglyph` *<picture>*, the last one being a synonym of `mplibdrawglyph` command.

An example:

```
mplibfillandstrokeglyph
  mplibglyph "R" of \fontid\font scaled 1/12
  withpen pencircle scaled 1
  withmplibcolors ("orange", 2/3red) ;
```



1.2.11 `mpliboutlinetext (...)`

As said before at § 1.1.3, `luamplib` provides the METAPOST operator `mpliboutlinetext` (*<string>*) which mimicks *metafun*'s `outlinetext`, but with some enhancements including the support for right-to-left writing direction. The syntax is the same as that of *metafun*: see the *metafun* documentation § 8.7 (texdoc *metafun*).

A simple example:

```
draw mpliboutlinetext.b ("$\sqrt{2+\alpha}$")
  (withcolor \mpcolor{red!50})
  (withpen pencircle scaled .25 withcolor 2/3red)
  scaled 3 ;
```



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 outline paths of 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.

`cff.lua` from ConT_EXt mkiv. As you see, `mplibglyph` operator requires `luaotfload` package loaded, which however is done automatically by \LaTeX format.

¹³*metafun* provides macros `nofill`, `eofill`, `fillup`, `eofillup` etc. (see *metafun* manual § 2.11), which `luamplib` with *plain* format does not provide currently.

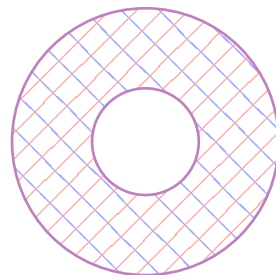
1.2.12 `\mppattern{...} ... \endmppattern, ... withmppattern ...`

TeX macros `\mppattern{<name>} ... \endmppattern` define a tiling pattern cell associated with the `<name>`. METAPOST command `withmppattern`, the syntax being `<cyclic path> | <textual picture> withmppattern <string>`, will fill the given path or text with the tiling pattern cell of the `<name>` by replicating it horizontally and vertically.¹⁴ As said before at § 1.2.6, the *textual picture* here means any text typeset by TeX, mostly the result of the `btex` command (and its derivatives) or the `infont` operator.

An example:

```
\mppattern{mypatt}           % or \begin{mppattern}{mypatt}
[                             % options: see below
  xstep = 10,
  ystep = 7,
  matrix = "rotated 45",      % or "0.7 0.7 -0.7 0.7" or {0.7, 0.7, -0.7, 0.7}
]
\mpfig                       % or any other TeX code
  draw (up--down) scaled 5
    withcolor 2/3[blue,white] ;
  draw (left--right) scaled 5
    withcolor 2/3[red,white] ;
\endmpfig
\endmppattern                % or \end{mppattern}

\mpfig
  mplibdrawglyph image(
    draw fullcircle scaled 100;
    draw reverse fullcircle scaled 40;
  )
  withmppattern "mypatt"
  withpen pencircle scaled 1
  withcolor \mpcolor{red!50!blue!50} ;
\endmpfig
```



The available options, actually elements of a Lua *table*, are listed in Table 1. For the sake of convenience, the width and height values of the tiling pattern cell 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 the 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 effect such as transparency in a pattern cell, `resources` option is needed: for instance, `resources="/ExtGState <<MyObj 5 0 R>>"`. However, as `luamplib` automatically includes the resources of the current page, this option is not needed in most cases.

¹⁴`withpattern` is an operator virtually the same as `withmppattern`, but the former forces a METAPOST picture. Therefore you cannot but use `draw` command with `withpattern` operator. On the other hand, `<cyclic path> withmppattern <string>` works as intended only with `fill` or `filldraw` command.

Table 1: options for \mppattern

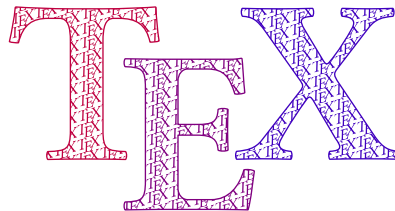
Key	Value Type	Explanation
xstep	<i>number</i>	horizontal spacing between pattern cells
ystep	<i>number</i>	vertical spacing between pattern cells
xshift	<i>number</i>	horizontal shifting of pattern cells
yshift	<i>number</i>	vertical shifting of pattern cells
bbox	<i>table</i> or <i>string</i>	llx, 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
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 cell which shall have no color at all (i.e. `withoutcolor` command is needed for METAPOST code).¹⁵ Uncolored pattern will be painted later by the color of a METAPOST object. An example:

```
\begin{mppattern}{pattnocolor}
[
  colored = false,
  matrix = "slanted .3 rotated 15",
]
\tiny\TeX
\end{mppattern}

\begin{mplibcode}
beginfig(1)
  picture tex;
  tex = mpliboutlinetext ("bfseries \TeX");
  for i=1 upto mpliboutlinenum:
    mplibfillandstrokeglyph mpliboutlinepic[i]
      scaled 8
      withmppattern "pattnocolor"
      withpen pencircle scaled 1/2
      withcolor (i/4)[red,blue]      % paints the pattern
    ;
  endfor
endfig;
\end{mplibcode}
```



A much simpler and efficient way to obtain a similar result (but 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"
```

¹⁵When using DVI mode, `-c` option might be needed to the `dvipdfmx` command.

```

fakebold 1/2
rotated 10 scaled 8
withmppattern "pattnocolor"
withmplibcolors (
    2/3[red,white],
    2/3 red
) ;
endfig;
\end{mplibcode}

```



1.2.13 ... asgroup ...

As said [before](#) at § 1.1.3, transparency group is available with *plain* as well as *metafun*. 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 syntax is basically the same as *metafun*'s: $\langle picture \rangle \mid \langle path \rangle \text{asgroup} \langle string \rangle$, the latter being `""` | `"isolated"` | `"knockout"` | `"isolated,knockout"` | `"off"`, which will return a METAPOST picture. The additional features provided by *luamplib* are:

- As shown, in addition to those arguments mimicking *metafun*'s, we allow another argument at the right-hand side: `asgroup "off"` will produce an ordinary *form XObject* rather than a transparency group *XObject*. On the contrary, `asgroup ""` (empty string) will produce a transparency group in which both of the PDF keys `/I` and `/K` are false.
- 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 bounding box 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 $(\langle pair \rangle, \langle pair \rangle)$ 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.

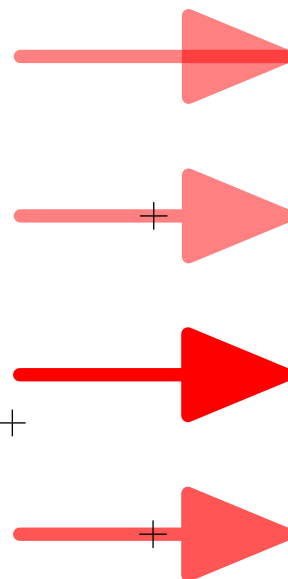
An example showing the effect of transparency group and the difference between the \TeX and METAPOST commands:

```
\mpfig
  picture pic;
  pic = image(drawarrow (left--right) scaled 5 withcolor red) scaled 10 ;
  draw pic
    asgroup "off"
    withtransparency (1, 1/2) ;
\endmpfig
```

```
\mpfig
  draw pic
    asgroup ""
    withgroupname "mygroup"
    withtransparency (1, 1/2) ;
  draw (left--right) scaled 5 ;
  draw (up--down) scaled 5 ;
\endmpfig
```

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

```
\mpfig
  usemplibgroup "mygroup"
  withtransparency (1, 2/3) ;
  draw (left--right) scaled 5 ;
  draw (up--down) scaled 5 ;
\endmpfig
```



Also note that normally the 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.

N.B. When you give shading effect upon a *textual picture* (ie. non-path object) inside or outside a transparency group, currently many of the PDF renderers, including Mac OS Preview and Foxit Editor, do not interpret PDF coordinates properly. If that happens, consider using other PDF viewer such as Adobe Acrobat. An example:

```
\mpfig*
  picture pic[];
  pic1 = image(
    fill fullcircle scaled 60 withoutcolor;
    fill fullcircle scaled 60 shifted 25right withoutcolor;
  ) ;
  pic2 = image(
    draw pic1
    withshadingmethod "linear"
```

```

        withshadingvector (2, 1)
        withshadingcolors (red, blue)
    ) ;
\endmpfig

```

```

\mpfig                                % shading inside group
draw pic2
  asgroup ""
  withtransparency (1, 2/3) ;
\endmpfig

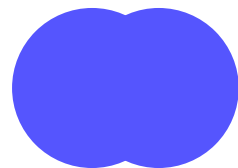
```



```

\mpfig                                % shading outside group
draw pic1
  asgroup ""
  withshadingmethod "linear"
  withshadingvector (2, 1)
  withshadingcolors (red, blue)
  withtransparency (1, 2/3) ;
\endmpfig

```

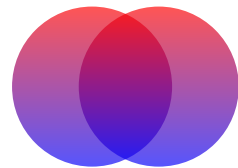


After some experiment, however, it turned out that the best way to get picture shading with transparency group is to give shading effect within an ordinary form XObject and then wrap it in a transparency group XObject. Most PDF renderers do render it properly:

```

\mpfig                                % shading within ordinary xobject
draw pic2
  asgroup "off"
  withgroupname "myshading"
  withtransparency (1, 2/3) ;
\endmpfig

```



```

\mpfig                                % wrap in a transparency group
usemplibgroup "myshading"
  asgroup ""
  withtransparency (1, 2/3) ;
\endmpfig

```



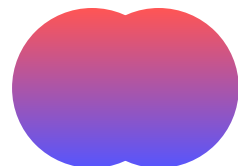
or preferably (see next subsection § 1.2.14 on \mplibgroup):

```

\mplibgroup{myshading}[asgroup="off"] % or no option
\mpfig
draw pic2 ;
\endmpfig
\endmplibgroup

\mpfig
usemplibgroup "myshading"
  asgroup ""
  withtransparency (1, 2/3) ;
\endmpfig

```



1.2.14 `\mplibgroup{...} ... \endmplibgroup`

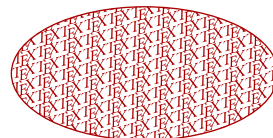
These TeX macros are described here in this subsection, as they are deeply related to the `asgroup` operator described just above at § 1.2.13. Users can define a transparency group or an ordinary form *XObject* with these macros from TeX side. The syntax is similar to the `\mppattern` command (see above § 1.2.12).

An example:¹⁶

```
\mplibgroup{texpatt1}           % or \begin{mplibgroup}{texpatt1}
[                               % options: see below
  asgroup="off",
]
\mpfig                         % or any other TeX code
  filldraw fullcircle xscaled 100 yscaled 50
    withmppattern "pattnocolor"
    withcolor 2/3 red ;
\endmpfig
\endmplibgroup                 % or \end{mplibgroup}
```

```
\mplibgroup{texpatt2}
[
  asgroup="",
]
\usemplibgroup{texpatt1}
\endmplibgroup

\usemplibgroup{texpatt2}
```



```
\mpfig
  usemplibgroup "texpatt2"
  rotated -15 scaled 1.2
  withtransparency (1, 2/3) ;
\endmpfig
```



Available options 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 is not given or is given as "off", an ordinary 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, just like the first figure in the example above at § 1.2.13.

As for the option `asgroup="masking"`, see the next subsection § 1.2.15.

With `colorspace` option, which is no-op for ordinary form *XObject*, users can specify the color space of a transparency group. For instance, the `mplibgroup` will be painted in grayscale model when `colorspace="/DeviceGray"` is given to an *isolated* transparency group.¹⁷

¹⁶Note that, here again, within a transparency group *XObject* a tiling pattern is encapsulated in an inner, ordinary *XObject*. This annoyance is especially for Mac OS Preview which misapplies the transformation matrix to tiling or shading patterns directly wrapped in a transparency group. Most other PDF renderers seem to behave properly even with single wrapping in a transparency group.

¹⁷Note that some PDF renderers such as Mac OS Preview or Firefox do not render properly this option.

Table 2: options for \mplibgroup

Key	Value Type	Explanation
asgroup	<i>string</i>	"" , "isolated", "knockout", "isolated,knockout", "off" or "masking"
colorspace	<i>string</i>	/CS entry to a transparency group, such as "/DeviceGray"
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

As shown, you can reuse the mplibgroup using the T_EX command \usemplibgroup or the METAPOST command usemplibgroup. The behavior of these commands is the same as that described above at § 1.2.13, excepting that the mplibgroup made by T_EX code (not by METAPOST code) respects original height and depth.

1.2.15 ... withmaskinggroup ...

Using this command, the mplibgroup (see above § 1.2.14) generated by the option asgroup="masking" (see Table 2) can be utilized as a masking transparency group upon a picture or a path object. The syntax is $\langle picture \rangle$ | $\langle path \rangle$ withmaskinggroup $\langle string \rangle$, the latter being the name of a pre-defined masking group.

Basically, the masking group should be prepared in *grayscale* color model: the area painted with 1 (\approx white: full luminosity) will preserve the full color of the object; the area painted with 0 (\approx black: zero luminosity) will force full transparency, masking it invisibly.¹⁸

By default, the background color of a masking group is 0 (\approx black), which you can change by this macro:

withmaskingbgcolor $\langle color\ expr \rangle$ sets the background color of the masking group. 0 denotes full transparency (masking invisibly); 1, full color.¹⁹

An example:

```
\mpfig*
picture pic;
pic = image(
  fill fullcircle scaled 80 withcolor blue ;
  fill fullcircle scaled 80 shifted (25,0) withcolor green ;
  fill fullcircle scaled 80 shifted (50,0) withcolor red ;
);
\endmpfig
```

¹⁸In fact, colors in other color models are also allowed (such as white, black, red, green, blue). But they will be converted to grayscale model by the PDF renderer, so that "/DeviceGray" is the default value of colorspace option to a masking transparency group (see Table 2 at § 1.2.14).

¹⁹Color expressions in rgb or cmyk model are also allowed, in accordance with the option colorspace="/DeviceRGB" or colorspace="/DeviceCMYK" given to the masking transparency group. This however we do not recommend as the luminosity is difficult to understand intuitively.


```

\mplibgroup{mymask}[asgroup="masking"]
\mpfig
  label(TEX "\sffamily\bfseries\scshape\Huge Meta" scaled 2, center pic)
  withcolor 1 ;
\endmpfig
\endmplibgroup

\mpfig
  fill bbox pic
  withshadingmethod "linear"
  withshadingcolors (red, blue) ;
draw pic
  withmaskinggroup "mymask"
  withmaskingbgcolor 1/10
  withtransparency (1, 0.8) ;
\endmpfig

```



1.2.16 `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 installed.

1.2.17 `mplibsubstring ... of ...`, `mplibucsubstring ... of ...`

`mplibsubstring` *<pair>* of *<string>* 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` *<pair>* of *<string>* 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 installed.

1.3 Lua

1.3.1 `runscript ...`

A good many METAPOST macros described in this documentation have been implemented using the primitive `runscript`. With `runscript` *<string>*, 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 data 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 example will print false, 3.0, MetaPost and the knots and the cyclicity of the path unitsquare.

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

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

Of course, this sort of Lua code can also be run inside METAPOST code using runscript command. Again, of course you can access a METAPOST variable using your own TeX macro. For example:

```
\def\mpnumeric#1#2{\directlua{
  tex.sprint(tostring(luamplib.instances["#1"]:get_numeric"#2"))
}}
\mpnumeric{myinstance}{n}\relax
```

3.0

1.3.3 Lua function luamplib.process_mplibcode

Users can run 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 affect the process of process_mplibcode.

Table 3: elements in `luamplib` table (partial)

Key	Type	Related \TeX macro	Cf.
<code>codeinherit</code>	<i>boolean</i>	<code>\mplibcodeinherit</code>	§ 1.1.8
<code>everyendmplib</code>	<i>table</i>	<code>\everyendmplib</code>	§ 1.1.2
<code>everymplib</code>	<i>table</i>	<code>\everymplib</code>	§ 1.1.2
<code>getcachedir</code>	<i>function</i> ($\langle string \rangle$)	<code>\mplibcachedir</code>	§ 1.1.15
<code>globaltexttext</code>	<i>boolean</i>	<code>\mplibglobaltexttext</code>	§ 1.1.9
<code>legacyverbatimtex</code>	<i>boolean</i>	<code>\mpliblegacybehavior</code>	§ 1.1.6
<code>noneedtoreplace</code>	<i>table</i>	<code>\mplibmakenocache</code>	§ 1.1.15
<code>numbersystem</code>	<i>string</i>	<code>\mplibnumbersystem</code>	§ 1.1.4
<code>setformat</code>	<i>function</i> ($\langle string \rangle$)	<code>\mplibsetformat</code>	§ 1.1.3
<code>showlog</code>	<i>boolean</i>	<code>\mplibshowlog</code>	§ 1.1.5
<code>texttextlabel</code>	<i>boolean</i>	<code>\mplibtexttextlabel</code>	§ 1.1.7
<code>verbatiminput</code>	<i>boolean</i>	<code>\mplibverbatim</code>	§ 1.1.11

1.3.4 Lua function `luamplib.registerpattern`

This is the Lua interface for `\mppattern ... \endmppattern` described above at § 1.2.12.

```
luamplib.registerpattern (<number> box register, <string> pattern name, <table> options)
```

The first argument is the register of a box containing a pattern cell, which should be prepared in advance by the user. For instance, `\setbox0=\hbox{\tiny\TeX}`, or corresponding Lua code using `tex.setbox` function; then the argument should be `0`.²⁰

As for the third argument, see above Table 1. The argument cannot be absent, but can be an empty table, i.e. `{ }`.

1.3.5 Lua function `luamplib.registergroup`

This is the Lua interface for `\mplibgroup ... \endmplibgroup` described above at § 1.2.14.

```
luamplib.registergroup (<number> box register, <string> group name, <table> options)
```

The first argument is the register of a box prepared in advance by the user. When the contents of the box have been generated from \TeX (not `METAPOST`) code, please make sure that both of the \TeX macros ‘`MP11x`’ and ‘`MP11y`’ are defined as ‘`0`’ before invoking the Lua function.²¹

As for the third argument, see above Table 2. The argument cannot be absent, but can be an empty table, i.e. `{ }`.

Reusing an `mplibgroup`, `\usemplibgroup{<name>}`, is basically the same as running the \TeX macro ‘`luamplib.group.<name>`’. If you need the `boxresource` index, inspect this macro using `token.get_macro` function.

²⁰In DVI mode, \TeX macro ‘`mplibpatternname`’ should be set as $\langle pattern name \rangle$ before preparing the box, if shading pattern (ie. shading on picture) is used in the pattern cell.

²¹In DVI mode, \TeX macro ‘`mplibgroupname`’ also should be set as $\langle group name \rangle$ before preparing the box, if shading pattern (ie. shading on picture) is used in the `mplibgroup`.

2 Implementation

2.1 Lua module

```
1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.41.0",
5   date      = "2026/04/28",
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. `ConTeXt` uses `metapost`.

```
9 luamplib      = luamplib or { }
10 local luamplib = luamplib
11
12 local format, abs = string.format, math.abs
13
14 Use our own function for warn/info/err.
15 local function termorlog (target, text, kind)
16   if text then
17     local mod, write, append = "luamplib", texio.write_nl, texio.write
18     kind = kind
19       or target == "term" and "Warning (more info in the log)"
20       or target == "log" and "Info"
21       or target == "term and log" and "Warning"
22       or "Error"
23     target = kind == "Error" and "term and log" or target
24     local t = text:explode"\n+"
25     write(target, format("Module %s %s:", mod, kind))
26     if #t == 1 then
27       append(target, format(" %s", t[1]))
28     else
29       for _,line in ipairs(t) do
30         write(target, line)
31       end
32       write(target, format("(%s) ", mod))
33     end
34     append(target, format(" on input line %s", tex.inputlineno))
35     write(target, "")
36     if kind == "Error" then error() end
37   end
38 end
39 local function warn (...) -- beware '%' symbol
40   termorlog("term and log", select("#",...) > 1 and format(...) or ...)
41 end
42 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

```

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 iioopen       = 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 = iioopen(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.mkdir or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\w/]+)") 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.

First of all, determine the directory to store cache files.

```

93 local cachedir
94 local function outputdir ()
95   if lfstouch then
96     for i,v in ipairs{'TEXMFVAR','TEXMF_OUTPUT_DIRECTORY','.', 'TEXMFOUTPUT'} do
97       local var = i == 3 and v or kpse.var_value(v)
98       if var and var ~= "" then
99         for _,vv in ipairs(var:explode(os.type == "unix" and ":" or ";")) do
100           local dir = format("%s/%s",vv,"luamplib_cache")
101           if not lfsisdir(dir) then
102             mk_full_path(dir)
103           end
104           if is_writable(dir) then
105             cachedir = dir; return cachedir
106           end
107         end
108       end
109     end
110   end
111   cachedir = "."; return cachedir
112 end
113 function luamplib.getcachedir(dir)
114   dir = dir:gsub("###","#")
115   dir = dir:gsub("^~",
116     os.type == "windows" and os.getenv("UserProfile") or os.getenv("HOME"))
117   if lfstouch and dir then
118     if lfsisdir(dir) then
119       if is_writable(dir) then
120         cachedir = dir
121       else
122         warn("Directory '%s' is not writable!", dir)
123       end
124     else
125       warn("Directory '%s' does not exist!", dir)
126     end
127   end
128 end

```

Some basic METAPOST files not necessary to make cache files.

```

129 local noneedtoreplace = {
130   ["boxes.mp"] = true, -- ["format.mp"] = true,
131   ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,

```

```

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

```

Pattern formats to replace btex and verbatimex ... etex in input files, if needed.

```

147 local name_b = "%f[%a_]"
148 local name_e = "%f[^%a_]"
149 local btex_etex = name_b.."btex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
150 local verbatimex_etex = name_b.."verbatimex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
151

```

Function luamplib.finder

```

152 local currenttime = os.time()
153 do
154   local luamplibtime = lfsattributes(kpse.find_file"luamplib.lua", "modification")

```

format.mp is much complicated, so specially treated.

```

155   local function replaceformatmp(file,newfile,ofmodify)
156     local fh = ioopen(file,"r")
157     if not fh then return file end
158     local data = fh:read("*all"); fh:close()
159     fh = ioopen(newfile,"w")
160     if not fh then return file end
161     fh:write(
162       "let normalinfont = infont;\n",
163       "primarydef str infont name = rawtexttext(str) enddef;\n",
164       data,
165       "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
166       "vardef Fexp_(expr x) = rawtexttext(\"$^{\"&decimal x&\"}$\") enddef;\n",
167       "let infont = normalinfont;\n"
168     ); fh:close()
169     lfstouch(newfile,currenttime,ofmodify)
170     return newfile
171   end
172   local function replaceinputmpfile (name,file)
173     local ofmodify = lfsattributes(file,"modification")
174     if not ofmodify then return file end
175     local newfile = name:gsub("%W","_")
176     newfile = format("%s/luamplib_input_%s", cachedir or outputdir(), newfile)

```

```

177   if newfile and luamplibtime then
178     local nf = lfsattributes(newfile)
179     if nf and nf.mode == "file" and
180       ofmodify == nf.modification and luamplibtime < nf.access then
181       return nf.size == 0 and file or newfile
182     end
183   end
184   if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
185   local fh = ioopen(file,"r")
186   if not fh then return file end
187   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.

```

188   local count,cnt = 0,0
189   data, cnt = data:gsub(btex_etex, "btex %1 etex ") -- space
190   count = count + cnt
191   data, cnt = data:gsub(verbatimt看etex, "verbatim %1 etex;") -- semicolon
192   count = count + cnt
193   if count == 0 then
194     needtoreplace[name] = true
195     fh = ioopen(newfile,"w");
196     if fh then
197       fh:close()
198       lfstouch(newfile,currenttime,ofmodify)
199     end
200     return file
201   end
202   fh = ioopen(newfile,"w")
203   if not fh then return file end
204   fh:write(data); fh:close()
205   lfstouch(newfile,currenttime,ofmodify)
206   return newfile
207 end

```

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.

```

208 local mpkpse
209 do
210   local exe = 0
211   while arg[exe-1] do
212     exe = exe-1
213   end
214   mpkpse = kpse.new(arg[exe], "mpost")
215 end
216 local special_ftype = {
217   pfb = "type1 fonts",
218   enc = "enc files",
219 }
220 function luamplib.finder (name, mode, ftype)

```



```

221   if mode == "w" then
222     if name and name ~= "mpout.log" then
223       kpse.record_output_file(name) -- recorder
224     end
225     return name
226   else
227     ftype = special_ftype[ftype] or ftype
228     local file = mpkpse.find_file(name, ftype)
229     if file then
230       if lfstouch and ftype == "mp" and not noneedtoreplace[name] and not noneedtoreplace["*.mp"] then
231         file = replaceinputmpfile(name, file)
232       end
233     else
234       file = mpkpse.find_file(name, name:match("%a+$"))
235     end
236     if file then
237       kpse.record_input_file(file) -- recorder
238     end
239     return file
240   end
241 end
242 end
243

```

For the main function: process

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

```

244 local currentformat = "plain"
245 function luamplib.setformat (name)
246   currentformat = name
247 end

```

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

```

248 luamplib.codeinherit = false
249 local mplibinstances = {}
250 luamplib.instances = mplibinstances
251 local has_instancename = false
252
253 local process
254 do
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. 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 loading plain/metafun format.

```

299 local preamble = tableconcat{
300     format(luamplib.preambles.preamble, replacesuffix(name,"mp")),
301     luamplib.preambles.mplibcode,
302     luamplib.legacyverbatim and luamplib.preambles.legacyverbatim 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 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.legacyverbatimtex),
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 end

```

353

dvipdfmx is supported, though nobody seems to use it.

```
354 local pdfmode = tex.outputmode > 0
```

355

make_text and some run_script uses LuaTeX's tex.runtoks.

```
356 local catlatex = luatexbase.registernumber("catcodetable@latex")
```

```
357 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.

```
358 local function run_tex_code (str, cat)
```

```
359   texruntoks(function() textsprint(cat or catlatex, str) end)
```

```
360 end
```

For conversion of sp to bp.

```
361 local factor = 65536*(7227/7200)
```

362

Prepare text 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.

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

```
364 local process_tex_text
```

```
365 do
```

```
366   local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
```

```
367     xscaled %f yscaled %f shifted (0,-%f) \z
```

```
368     withprescript "mplibtexboxid=%i:%f:%f")'
```

```
369   function process_tex_text (str, maketext)
```

```
370     if str then
```

```
371       if not maketext then str = str:gsub("\r.-$", "") end
```

```
372       local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
```

```
373         and "\\global" or ""
```

```
374       local tex_box_id
```

```
375       if global == "" then
```

```
376         tex_box_id = texboxes.localid + 1
```

```
377         texboxes.localid = tex_box_id
```

```
378       else
```

```
379         local boxid = texboxes.globalid + 1
```

```
380         texboxes.globalid = boxid
```

```
381         run_tex_code(format([[\\expandafter\\newbox\\csname luamplib.box.%s\\endcsname]], boxid))
```

```
382         tex_box_id = tex.getcount'allocationnumber'
```

```
383       end
```

```
384       if str:find"^[taggingoff%]" then
```

```
385         str = str:gsub("^[taggingoff%]s*", "")
```

```
386         run_tex_code(format("\\luamplibnotagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
```

```
387           tex_box_id, global, tex_box_id, str))
```

```
388       else
```

```
389         run_tex_code(format("\\luamplibtagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
```

```

390             tex_box_id, global, tex_box_id, str))
391     end
392     local box = texgetbox(tex_box_id)
393     local wd = box.width / factor
394     local ht = box.height / factor
395     local dp = box.depth / factor
396     return text_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
397 end
398 return ""
399 end
400 end
401

```

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.

```

402 if is_defined'color_select:n' then
403   run_tex_code{
404     "\newcatcodetable\luamplibcctabexplat",
405     "\begingroup",
406     "\catcode\@=11 ",
407     "\catcode\_ =11 ",
408     "\catcode\:=11 ",
409     "\savecatcodetable\luamplibcctabexplat",
410     "\endgroup",
411   }
412 end
413 local ccexplat = luatexbase.registernumber"luamplibcctabexplat"
414
415 local process_color, process_mplibcolor

```

A common function for color functions

```

416 local function colorsplit (res)
417   local t, tt = { }, res:gsub("[%[%]]", "", 2):explode()
418   local be = tt[1]:find"^%d" and 1 or 2
419   for i=be, #tt do
420     if not tonumber(tt[i]) then break end
421     t[#t+1] = tt[i]
422   end
423   if #t == 0 then -- named color in DVI mode with no DocumentMetadata
424     run_tex_code{"\extractcolourspecs{", tt[3], "}\mplibtmpa\mplibtmpb"}
425     t = get_macro"mplibtmpb":explode",
426   end
427   return t
428 end
429 do
430   local colfmt = ccexplat and "l3color" or "xcolor"
431   local mplibcolorfmt = {
432     xcolor = tableconcat{
433       [[\begingroup\let\XC@color\relax]],
434       [[\def\set@color{\global\mplibtmp toks\expandafter{\current@color}}]],

```

```

435     [[\color%s\endgroup]],
436 },
437 l3color = tableconcat{
438     [[\beginpgroup\def\__color_select:N#1{\expandafter\__color_select:nn#1}]],
439     [[\def\__color_backend_select:nn#1#2{\global\mplibtmptoks{#1 #2}}]],
440     [[\def\__kernel_backend_literal:e#1{\global\mplibtmptoks\expandafter{\expanded{#1}}}],
441     [[\color_select:n%s\endgroup]],
442 },
443 }
444 function process_color (str)
445     if str then
446         if not str:find("%b{") then
447             str = format("{%s}",str)
448         end
449         local myfmt = mplibcolorfmt[colfmt]
450         if colfmt == "l3color" and is_defined"color" then
451             if str:find("%b[") then
452                 myfmt = mplibcolorfmt.xcolor
453             else
454                 for _,v in ipairs(str:match"{{(.+)}}":explode"!") do
455                     if not v:find("^%s*%d+%s*$") then
456                         local pp = get_macro(format("l__color_named_%s_prop",v))
457                         if not pp or pp == "" then
458                             myfmt = mplibcolorfmt.xcolor
459                             break
460                         end
461                     end
462                 end
463             end
464         end
465         run_tex_code(myfmt:format(str), ccexplat or catat11)
466         local t = texgettoks"mplibtmptoks"
467         if not pdfmode then
468             if t:find"^hsb" or not t:find"%d" then
469                 t = "color push " .. t
470             elseif not t:find"^pdf" then
471                 t = t:gsub("%a+ (.+)", "pdf:bc [%1]")
472             end
473         end
474         return format('1 withprescript "mpliboverridecolor=%s"', t)
475     end
476     return ""
477 end
478 function process_mplibcolor(str)
479     local res = process_color(str)
480     if res:find" cs " or res:find"@pdf.obj" or res:find"color push" then return res end
481     res = colorsplit(res:match"mpliboverridecolor=(.+)")
482     return format("(%s)", tableconcat(res, ","))
483 end

```

```

484 end
485
    for \mpdim or mplibdimen
486 local function process_dimen (str)
487   if str then
488     str = str:gsub("{(.+)}", "%1")
489     run_tex_code(format([[\\mplibtmp toks\\expandafter{\\the\\dimexpr %s\\relax}]], str))
490     return format("begingroup %s endgroup", texgettoks"mplibtmp toks")
491   end
492   return ""
493 end
494

```

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

```

495 local function process_verbatimtex_text (str)
496   if str then
497     run_tex_code(str)
498   end
499   return ""
500 end
501

```

For legacy verbatimtex process. verbatimtex ... etex before `beginfig()` is inserted just before the `mplib` box. And `TEX` code inside `beginfig()` ... `endfig` is inserted after the `mplib` box.

```

502 local tex_code_pre_mplib = {}
503 luamplib.figid = 1
504 luamplib.in_the_fig = false
505 local function process_verbatimtex_prefig (str)
506   if str then
507     tex_code_pre_mplib[luamplib.figid] = str
508   end
509   return ""
510 end
511 local function process_verbatimtex_infig (str)
512   if str then
513     return format('special "postmplibverbtx=%s";', str)
514   end
515   return ""
516 end
517

```

For *metafun* format. see issue #79.

```

518 mp = mp or {}
519 local mp = mp
520 mp.mf_path_reset = mp.mf_path_reset or function() end
521 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
522 mp.report = mp.report or info

```

metafun 2021-03-09 changes crashes luamplib.

```

523 catcodes = catcodes or {}

```

```

524 local catcodes = catcodes
525 catcodes.numbers = catcodes.numbers or {}
526 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
527 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
528 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
529 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
530 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
531 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
532 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
533

```

Now `luamplib.runscript`

```

534 do
535   local runscript_funcs = {
536     luamplibtext    = process_tex_text,
537     luamplibcolor   = process_mplibcolor,
538     luamplibdimen   = process_dimen,
539     luamplibprefig  = process_verbatimtex_prefig,
540     luamplibinfig   = process_verbatimtex_infig,
541     luamplibverbtex = process_verbatimtex_text,
542   }

```

A function from ConTeXt general.

```

543   local function mpprint(buffer,...)
544     for i=1,select("#",...) do
545       local value = select(i,...)
546       if value ~= nil then
547         local t = type(value)
548         if t == "number" then
549           buffer[#buffer+1] = format("%.16f",value)
550         elseif t == "string" then
551           buffer[#buffer+1] = value
552         elseif t == "table" then
553           buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
554         else -- boolean or whatever
555           buffer[#buffer+1] = tostring(value)
556         end
557       end
558     end
559   end
560   function luamplib.runscript (code)
561     local id, str = code:match("(.-){(.*)}")
562     if id and str then
563       local f = runscript_funcs[id]
564       if f then
565         local t = f(str)
566         if t then return t end
567       end
568     end
569     local f = loadstring(code)

```



```

570   if type(f) == "function" then
571     local buffer = {}
572     function mp.print(...)
573       mpprint(buffer,...)
574     end
575     local res = {f()}
576     buffer = tableconcat(buffer)
577     if buffer and buffer ~= "" then
578       return buffer
579     end
580     buffer = {}
581     mpprint(buffer, tableunpack(res))
582     return tableconcat(buffer)
583   end
584   return ""
585 end
586 end
587

```

luamplib.maketext

```

588 luamplib.legacyverbatimtex = true
589 do

```

make_text must be one liner, so comment sign is not allowed.

```

590   local function protecttexcontents (str)
591     return str:gsub("\\%", "\\0PerCent\0")
592           :gsub("%%.\n", "")
593           :gsub("%%.-$", "")
594           :gsub("%zPerCent%z", "\\%")
595           :gsub("\r.-$", "")
596           :gsub("%s+", " ")
597   end
598   function luamplib.maketext (str, what)
599     if str and str ~= "" then
600       str = protecttexcontents(str)
601       if what == 1 then
602         if not str:find("\\documentclass"..name_e) and
603            not str:find("\\begin%s*{document}") and
604            not str:find("\\documentstyle"..name_e) and
605            not str:find("\\usepackage"..name_e) then
606           if luamplib.legacyverbatimtex then
607             if luamplib.in_the_fig then
608               return process_verbatimtex_infig(str)
609             else
610               return process_verbatimtex_prefig(str)
611             end
612           else
613             return process_verbatimtex_text(str)
614           end
615         end

```

```

616     else
617         return process_tex_text(str, true) -- bool is for 'char13'
618     end
619 end
620 return ""
621 end
622 end
623

```

luamplib's METAPOST color operators

```

624 luamplib.gettexcolor = function (str, rgb)
625     local res = process_color(str):match'"mpliboverridecolor=(.)"'
626     if res:find" cs " or res:find"@pdf.obj" then
627         if not rgb then
628             warn("%s is a spot color. Forced to CMYK", str)
629         end
630         run_tex_code({
631             "\\color_export:nnN{",
632             str,
633             "}{",
634             rgb and "space-sep-rgb" or "space-sep-cmyk",
635             "}"\\mplib_@tempa",
636         },ccexplat)
637         return get_macro"mplib_@tempa":explode()
638     end
639     local t = colorsplit(res)
640     if #t == 3 or not rgb then return t end
641     if #t == 4 then
642         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]) }
643     end
644     return { t[1], t[1], t[1] }
645 end
646
647 luamplib.shadecolor = function (str)
648     local res = process_color(str):match'"mpliboverridecolor=(.)"'
649     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}

650   run_tex_code({
651     [[\color_export:nnN{]], str, [[{backend}\mplib@tempa]],
652   },ccexplat)
653   local name, value = get_macro'mplib@tempa':match'{{(.-)}}{(.-)}'
654   local t, obj = res:explode()
655   if pdfmode then
656     obj = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
657   else
658     obj = t[2]
659   end
660   return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
661 end
662 return colorsplit(res)
663 end
664

luamplib.fillandstrokecolor
665 do
666   local function graphictextcolor (col, filldraw)
667     if col:find"^[%d%.:]+$" then
668       col = col:explode":"
669       for i=1,#col do
670         col[i] = format("%.3f", col[i])
671       end
672       if pdfmode then
673         local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
674         col[#col+1] = filldraw == "fill" and op or op:upper()
675         return tableconcat(col," ")
676       end
677       return format("[%s]", tableconcat(col," "))
678     end
679     col = process_color(col):match'"mpliboverridecolor=(.+)"'
680     if pdfmode then
681       local t = col:explode()
682       local b = filldraw == "fill" and 1 or #t/2+1

```

```

683     local e = b == 1 and #t/2 or #t
684     return tableconcat(t, " ", b, e)
685 end
686 if col:find"@pdf.obj" then
687     return col:gsub("pdf:bc%s*", "", 1)
688 else
689     return format("[%s]", tableconcat(colorsplit(col), " "))
690 end
691 end
692 function luamplib.fillandstrokecolor (fill, stroke)
693     fill = graphictextcolor(fill, "fill")
694     stroke = graphictextcolor(stroke, "stroke")
695     local bc = pdfmode and "" or "pdf:bc "
696     return format('withprescript "mpliboverridecolor=%s%s %s"', bc, fill, stroke)
697 end
698 end
699

```

Remove trailing zeros for smaller PDF

```

700 local decimals = "%.d+"
701 local function rmzeros(str) return str:gsub("%.?0+$", "") end
702

```

common function for mplibgraphictext and mpliboutlinetext

```

703 local function getrulemetric (box, curr, bp)
704     local running = -1073741824
705     local wd,ht,dp = curr.width, curr.height, curr.depth
706     wd = wd == running and box.width or wd
707     ht = ht == running and box.height or ht
708     dp = dp == running and box.depth or dp
709     if bp then
710         return wd/factor, ht/factor, dp/factor
711     end
712     return wd, ht, dp
713 end
714

```

luamplib's mplibgraphictext operator

```

715 do
716     if not math.round then
717         function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
718     end
719     local emboldenfonts = { }
720     local function roundupwidth (f, fb)
721         local wd = math.round(f.size * fb / factor * 10)
722         if wd == 0 and fb ~= 0 then
723             wd = 1
724         end
725         emboldenfonts.width = wd
726         return wd

```

```

727 end
728 local function getemboldenwidth (curr, fakebold)
729     local width = emboldenfonts.width
730     if not width then
731         local f
732         local function getglyph(n)
733             while n do
734                 if n.head then
735                     getglyph(n.head)
736                 elseif n.font and n.font > 0 then
737                     f = n.font; break
738                 end
739                 n = node.getnext(n)
740             end
741         end
742         getglyph(curr)
743         width = roundupwidth(font.getcopy(f or font.current()), fakebold)
744     end
745     return width
746 end
747 local function getrulewhatsit (line, wd, ht, dp)
748     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
749     line = line == 0 and "" or ("%f w"):format(line)
750     local pl
751     local fmt = "q %s %f %f %f %f re B Q"
752     if pdfmode then
753         pl = node.new("whatsit", "pdf_literal")
754         pl.mode = 0
755     else
756         fmt = "pdf:content " .. fmt
757         pl = node.new("whatsit", "special")
758     end
759     pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
760     local ss = node.new"glue"
761     node.setglue(ss, 0, 65536, 65536, 2, 2)
762     pl.next = ss
763     return pl
764 end

```

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

```

765 local tag_update_attrs
766 if is_defined"ver@tagpdf.sty" then
767     tag_update_attrs = function (n, curr)
768         while n do
769             n.attr = curr.attr
770             if n.head then
771                 tag_update_attrs(n.head, curr)
772             end
773             n = node.getnext(n)

```

```

774     end
775   end
776   else
777     tag_update_attrs = function() end
778   end
779   local function embolden (box, curr, fakebold)
780     local head = curr
781     while curr do
782       if curr.head then
783         curr.head = embolden(curr, curr.head, fakebold)
784       elseif curr.replace then
785         curr.replace = embolden(box, curr.replace, fakebold)
786       elseif curr.leader then
787         if curr.leader.head then
788           curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
789         elseif curr.leader.id == node.id"rule" then
790           local glue = node.effective_glue(curr, box)
791           local line = getemboldenwidth(curr, fakebold)
792           local wd,ht,dp = getrulemetric(box, curr.leader)
793           if box.id == node.id"hlist" then
794             wd = glue
795           else
796             ht, dp = 0, glue
797           end
798           local pl = getrulewhatsit(line, wd, ht, dp)
799           local pack = box.id == node.id"hlist" and node.hpack or node.vpack
800           local list = pack(pl, glue, "exactly")
801           tag_update_attrs(list,curr)
802           head = node.insert_after(head, curr, list)
803           head, curr = node.remove(head, curr)
804         end
805       elseif curr.id == node.id"rule" and curr.subtype == 0 then
806         local line = getemboldenwidth(curr, fakebold)
807         local wd,ht,dp = getrulemetric(box, curr)
808         if box.id == node.id"vlist" then
809           ht, dp = 0, ht+dp
810         end
811         local pl = getrulewhatsit(line, wd, ht, dp)
812         local list
813         if box.id == node.id"hlist" then
814           list = node.hpack(pl, wd, "exactly")
815         else
816           list = node.vpack(pl, ht+dp, "exactly")
817         end
818         tag_update_attrs(list,curr)
819         head = node.insert_after(head, curr, list)
820         head, curr = node.remove(head, curr)
821       elseif curr.id == node.id"glyph" and curr.font > 0 then
822         local f = curr.font

```

```

823     local key = format("%s:%s",f,fakebold)
824     local i = emboldenfonts[key]
825     if not i then
826         local ft = font.getfont(f) or font.getcopy(f)
827         local width = roundupwidth(ft, fakebold)
828         if ft.format == "opentype" or ft.format == "truetype" then
829             local name = ft.name.gsub("'",'):gsub('$','')
830             local t = name.gsub("^file:","):gsub("^name:","):gsub("^kpse:","):gsub("^my:",")
831             name = format('%s%sembolden=%s;',name, t:find":" and ";" or ":", fakebold)
832             _, i = fonts.constructors.readanddefine(name,ft.size)
833         elseif pdfmode then
834             local ft = table.copy(ft)
835             ft.mode, ft.width = 2, width
836             i = font.define(ft)
837         else
838             goto skip_type1
839         end
840         emboldenfonts[key] = i
841     end
842     curr.font = i
843 end
844 ::skip_type1::
845     curr = node.getnext(curr)
846 end
847 return head
848 end
849 luamplib.graphicstext = function (text, fakebold, fc, dc)
850     local fmt = process_tex_text(text):sub(1,-2)
851     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
852     emboldenfonts.width = nil
853     local box = texgetbox(id)
854     box.head = embolden(box, box.head, fakebold)
855     local colors = luamplib.fillandstrokecolor(fc, dc)
856     return format('%s %s)', fmt, colors)
857 end
858 end
859

```

luamplib's mplibglyph operator

```

860 do
861     local function mperr (str)
862         return format("hide(errmessage %q)", str)
863     end
864     local function getangle (a,b,c)
865         local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
866         if r > 180 then
867             r = r - 360
868         elseif r < -180 then
869             r = r + 360

```



```

870     end
871     return r
872 end
873 local function turning (t)
874     local r, n = 0, #t
875     for i=1,2 do
876         tableinsert(t, t[i])
877     end
878     for i=1,n do
879         r = r + getangle(t[i], t[i+1], t[i+2])
880     end
881     return r/360
882 end
883 local function glyphimage(t, fmt)
884     local q, p, r, towarn = {},{}
885     local function closepath(dots)
886         tableinsert(p, format("%scycle", dots or "--"))
887         tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
888     end
889     for i,v in ipairs(t) do
890         local cmd = v[#v]
891         local nt = t[i+1]
892         local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
893         if cmd == "m" then
894             if final then towarn = true end
895             p = {format('(%s,%s)',v[1],v[2])}
896             r = {{x=v[1],y=v[2]}}
897         else
898             if cmd == "l" then
899                 local pt = t[i-1]
900                 if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
901                     else
902                         tableinsert(p, format('--(%s,%s)',v[1],v[2]))
903                         tableinsert(r, {x=v[1],y=v[2]})
904                     end
905                 if final then closepath() end
906             elseif cmd == "c" then
907                 tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
908                 if final and r[1].x == v[5] and r[1].y == v[6] then
909                     closepath ".."
910                 else
911                     tableinsert(p, format('..(%s,%s)',v[5],v[6]))
912                     tableinsert(r, {x=v[5],y=v[6]})
913                     if final then closepath() end
914                 end
915             elseif cmd == "path" or cmd == "move" then
916             else
917                 return mperr"unknown operator"
918             end

```

```

919     end
920 end
921 r = { }
922 if fmt == "opentype" then
923     for _,v in ipairs(q[1]) do
924         tableinsert(r, format('addto currentpicture contour %s;',v))
925     end
926     for _,v in ipairs(q[2]) do
927         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
928     end
929 else
930     for _,v in ipairs(q[2]) do
931         tableinsert(r, format('addto currentpicture contour %s;',v))
932     end
933     for _,v in ipairs(q[1]) do
934         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
935     end
936 end
937 return format('image(%s)', tableconcat(r)), towarn
938 end
939 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
940 function luamplib.glyph (f, c)
941     local filename, subfont, instance, kind, shapedata
942     local fid = tonumber(f) or font.id(f)
943     if fid > 0 then
944         local fontdata = font.getfont(fid) or font.getcopy(fid)
945         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
946         instance = fontdata.specification and fontdata.specification.instance
947             or fontdata.shared and fontdata.shared.features.axis
948         filename = filename and filename:gsub("^harfloaded:", "")
949     else
950         local name
951         f = f:match"^%s*(.)%s*$"
952         name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
953         if not name then
954             name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
955         end
956         if not name then
957             name, subfont = f:match"(.+)%((%d+)%)%" -- Times.ttc(2)
958         end
959         name = name or f
960         subfont = (subfont or 0)+1
961         instance = instance and instance:lower()
962         for _,ftype in ipairs{"opentype", "truetype"} do
963             filename = kpse.find_file(name, ftype.." fonts")
964             if filename then
965                 kind = ftype; break
966             end
967         end

```

```

968     end
969     if kind ~= "opentype" and kind ~= "truetype" then
970         f = fid and fid > 0 and tex.fontname(fid) or f
971         if kpse.find_file(f, "tfm") then
972             return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
973         else
974             filename = kpse.find_file(f, "type1 fonts")
975             if filename then
976                 kind = "type1" -- there's bug in processing cmr family
977             else
978                 return mperr"font not found"
979             end
980         end
981     end
982     local time = lfsattributes(filename,"modification")

    local k = format("shapes_%s(%s)[%s]%s", filename, subfont or "", instance or "",
        luaotfload and luaotfload.version or "")

983     local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
984     local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
985     local newname = format("%s/%s.lua", cachedir or outputdir(), h)
986     local newtime = lfsattributes(newname,"modification") or 0
987     if time == newtime then
988         shapedata = require(newname)
989     end
990     if not shapedata then
991         if fonts then
992             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
993             shapedata = handler.readers.loadshapes(filename,subfont,instance)
994         end
995         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
996         table.tofile(newname, shapedata, "return")
997         lfstouch(newname, time, time)
998     end
999     local gid = tonumber(c)
1000     if not gid then
1001         local uni = utf8.codepoint(c)
1002         for i,v in pairs(shapedata.glyphs) do
1003             if c == v.name or uni == v.unicode then
1004                 gid = i; break
1005             end
1006         end
1007     end
1008     if not gid then return mperr"cannot get GID (glyph id)" end
1009     local fac = 1000 / (shapedata.units or 1000)
1010     local t = shapedata.glyphs[gid]; t = t and t.segments
1011     if not t then return "image()" end
1012     for i,v in ipairs(t) do

```

```

1013     if type(v) == "table" then
1014         for ii,vv in ipairs(v) do
1015             if type(vv) == "number" then
1016                 t[i][ii] = format("%.0f", vv * fac)
1017             end
1018         end
1019     end
1020 end
1021 local result, towarn = glyphimage(t, shapedata.format or kind)
1022 if towarn then
1023     warn("mplibglyph %s not working properly. Use glyph instead", f)
1024 end
1025 return result
1026 end
1027 end
1028

```

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

```

1029 do
1030     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1031         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1032     local outline_horz, outline_vert
1033     function outline_vert (res, box, curr, xshift, yshift)
1034         local b2u = box.dir == "LTL"
1035         local dy = (b2u and -box.depth or box.height)/factor
1036         local ody = dy
1037         while curr do
1038             if curr.id == node.id"rule" then
1039                 local wd, ht, dp = getrulemetric(box, curr, true)
1040                 local hd = ht + dp
1041                 if hd ~= 0 then
1042                     dy = dy + (b2u and dp or -ht)
1043                     if wd ~= 0 and curr.subtype == 0 then
1044                         res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1045                     end
1046                     dy = dy + (b2u and ht or -dp)
1047                 end
1048             elseif curr.id == node.id"glue" then
1049                 local vwidth = node.effective_glue(curr,box)/factor
1050                 if curr.leader then
1051                     local curr, kind = curr.leader, curr.subtype
1052                     if curr.id == node.id"rule" then
1053                         local wd = getrulemetric(box, curr, true)
1054                         if wd ~= 0 then
1055                             local hd = vwidth
1056                             local dy = dy + (b2u and 0 or -hd)
1057                             if hd ~= 0 and curr.subtype == 0 then
1058                                 res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1059                             end

```

```

1060         end
1061     elseif curr.head then
1062         local hd = (curr.height + curr.depth)/factor
1063         if hd <= vwidth then
1064             local dy, n, iy = dy, 0, 0
1065             if kind == 100 or kind == 103 then -- todo: gleaders
1066                 local ady = abs(ody - dy)
1067                 local ndy = math.ceil(ady / hd) * hd
1068                 local diff = ndy - ady
1069                 n = math.floor((vwidth-diff) / hd)
1070                 dy = dy + (b2u and diff or -diff)
1071             else
1072                 n = math.floor(vwidth / hd)
1073                 if kind == 101 then
1074                     local side = vwidth % hd / 2
1075                     dy = dy + (b2u and side or -side)
1076                 elseif kind == 102 then
1077                     iy = vwidth % hd / (n+1)
1078                     dy = dy + (b2u and iy or -iy)
1079                 end
1080             end
1081             dy = dy + (b2u and curr.depth or -curr.height)/factor
1082             hd = b2u and hd or -hd
1083             iy = b2u and iy or -iy
1084             local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1085             for i=1,n do
1086                 res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1087                 dy = dy + hd + iy
1088             end
1089         end
1090     end
1091     end
1092     dy = dy + (b2u and vwidth or -vwidth)
1093     elseif curr.id == node.id"kern" then
1094         dy = dy + curr.kern/factor * (b2u and 1 or -1)
1095     elseif curr.id == node.id"vlist" then
1096         dy = dy + (b2u and curr.depth or -curr.height)/factor
1097         res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1098         dy = dy + (b2u and curr.height or -curr.depth)/factor
1099     elseif curr.id == node.id"hlist" then
1100         dy = dy + (b2u and curr.depth or -curr.height)/factor
1101         res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1102         dy = dy + (b2u and curr.height or -curr.depth)/factor
1103     end
1104     curr = node.getnext(curr)
1105 end
1106 return res
1107 end
1108 function outline_horz (res, box, curr, xshift, yshift, discwd)

```

```

1109 local r2l = box.dir == "TRT"
1110 local dx = r2l and (discwd or box.width/factor) or 0
1111 local dirs = { { dir = r2l, dx = dx } }
1112 while curr do
1113   if curr.id == node.id"dir" then
1114     local sign, dir = curr.dir:match"(.)(...)"
1115     local level, newdir = curr.level, r2l
1116     if sign == "+" then
1117       newdir = dir == "TRT"
1118       if r2l ~= newdir then
1119         local n = node.getnext(curr)
1120         while n do
1121           if n.id == node.id"dir" and n.level+1 == level then break end
1122           n = node.getnext(n)
1123         end
1124         n = n or node.tail(curr)
1125         dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1126       end
1127       dirs[level] = { dir = r2l, dx = dx }
1128     else
1129       local level = level + 1
1130       newdir = dirs[level].dir
1131       if r2l ~= newdir then
1132         dx = dirs[level].dx
1133       end
1134     end
1135     r2l = newdir
1136   elseif curr.char and curr.font and curr.font > 0 then
1137     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1138     local gid = ft.characters[curr.char].index or curr.char
1139     local scale = ft.size / factor / 1000
1140     local slant = (ft.slant or 0)/1000
1141     local extend = (ft.extend or 1000)/1000
1142     local squeeze = (ft.squeeze or 1000)/1000
1143     local expand = 1 + (curr.expansion_factor or 0)/1000000
1144     local xscale, yscale = scale * extend * expand, scale * squeeze
1145     dx = dx - (r2l and curr.width/factor*expand or 0)
1146     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1147     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1148     local vertical = ""
1149     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1150       if ft.shared.features.vertical then -- luatexko
1151         vertical = "rotated 90"
1152         local data = ft.characters[curr.char] or { }
1153         if ft.hb then
1154           local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1155           local charraise = (ft.luatexko_charraise or 0)/factor
1156           xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1157         else

```

```

1158         local cmds = data.commands or { {0,0}, {0,0} }
1159         local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1160         xpos, ypos = xpos + hoff, ypos + voff
1161     end
1162     elseif curr ~= box.head then -- luatexja
1163         vertical = "rotated 90"
1164         local en = ft.parameters.quad/factor/2
1165         xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1166     end
1167 end
1168 local image
1169 if ft.format == "opentype" or ft.format == "truetype" then
1170     image = luamplib.glyph(curr.font, gid)
1171 else
1172     local name, scale = ft.name, 1
1173     local vf = font.read_vf(name, ft.size)
1174     if vf and vf.characters[gid] then
1175         local cmds = vf.characters[gid].commands or {}
1176         for _,v in ipairs(cmds) do
1177             if v[1] == "char" then
1178                 gid = v[2]
1179             elseif v[1] == "font" and vf.fonts[v[2]] then
1180                 name = vf.fonts[v[2]].name
1181                 scale = vf.fonts[v[2]].size / ft.size
1182             end
1183         end
1184     end
1185     image = format("glyph %s of %q scaled %f", gid, name, scale)
1186 end
1187 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1188                     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1189 dx = dx + (r2l and 0 or curr.width/factor*expand)
1190 elseif curr.replace then
1191     local width = node.dimensions(curr.replace)/factor
1192     dx = dx - (r2l and width or 0)
1193     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1194     dx = dx + (r2l and 0 or width)
1195 elseif curr.id == node.id"rule" then
1196     local wd, ht, dp = getrulemetric(box, curr, true)
1197     if wd ~= 0 then
1198         local hd = ht + dp
1199         dx = dx - (r2l and wd or 0)
1200         if hd ~= 0 and curr.subtype == 0 then
1201             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1202         end
1203         dx = dx + (r2l and 0 or wd)
1204     end
1205 elseif curr.id == node.id"glue" then
1206     local width = node.effective_glue(curr, box)/factor

```

```

1207     dx = dx - (r2l and width or 0)
1208     if curr.leader then
1209         local curr, kind = curr.leader, curr.subtype
1210         if curr.id == node.id"rule" then
1211             local wd, ht, dp = getrulemetric(box, curr, true)
1212             local hd = ht + dp
1213             if hd ~= 0 then
1214                 wd = width
1215                 if wd ~= 0 and curr.subtype == 0 then
1216                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1217                 end
1218             end
1219         elseif curr.head then
1220             local wd = curr.width/factor
1221             if wd <= width then
1222                 local dx = r2l and dx+width or dx
1223                 local n, ix = 0, 0
1224                 if kind == 100 or kind == 103 then -- todo: gleaders
1225                     local adx = abs(dx-dirs[1].dx)
1226                     local ndx = math.ceil(adx / wd) * wd
1227                     local diff = ndx - adx
1228                     n = math.floor((width-diff) / wd)
1229                     dx = dx + (r2l and -diff-wd or diff)
1230                 else
1231                     n = math.floor(width / wd)
1232                     if kind == 101 then
1233                         local side = width % wd / 2
1234                         dx = dx + (r2l and -side-wd or side)
1235                     elseif kind == 102 then
1236                         ix = width % wd / (n+1)
1237                         dx = dx + (r2l and -ix-wd or ix)
1238                     end
1239                 end
1240                 wd = r2l and -wd or wd
1241                 ix = r2l and -ix or ix
1242                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1243                 for i=1,n do
1244                     res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1245                     dx = dx + wd + ix
1246                 end
1247             end
1248         end
1249     end
1250     dx = dx + (r2l and 0 or width)
1251 elseif curr.id == node.id"kern" then
1252     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1253 elseif curr.id == node.id"math" then
1254     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1255 elseif curr.id == node.id"vlist" then

```



```

1256     dx = dx - (r2l and curr.width/factor or 0)
1257     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1258     dx = dx + (r2l and 0 or curr.width/factor)
1259     elseif curr.id == node.id"hlist" then
1260         dx = dx - (r2l and curr.width/factor or 0)
1261         res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1262         dx = dx + (r2l and 0 or curr.width/factor)
1263     end
1264     curr = node.getnext(curr)
1265 end
1266 return res
1267 end
1268 function luamplib.outlinetext (text)
1269     local fmt = process_tex_text(text)
1270     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1271     local box = texgetbox(id)
1272     local res = outline_horz({ }, box, box.head, 0, 0)
1273     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1274     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1275 end
1276 end
1277

```

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

```

1278 function luamplib.getunicodegraphemes (s)
1279     local t = { }
1280     local graphemes = require'lua-uni-graphemes'
1281     for _, _, c in graphemes.graphemes(s) do
1282         table.insert(t, c)
1283     end
1284     return t
1285 end
1286 function luamplib.unicodesubstring (s,b,e,grph)
1287     local tt, t, step = { }
1288     if grph then
1289         t = luamplib.getunicodegraphemes(s)
1290     else
1291         t = { }
1292         for _, c in utf8.codes(s) do
1293             table.insert(t, utf8.char(c))
1294         end
1295     end
1296     if b <= e then
1297         b, step = b+1, 1
1298     else
1299         e, step = e+1, -1
1300     end
1301     for i = b, e, step do
1302         table.insert(tt, t[i])
1303     end
1304     return tt
1305 end

```

```

1303 end
1304 s = table.concat(tt):gsub("'", "'&ditto&'")
1305 return string.format("%s", s)
1306 end
1307

METAPOST preambles

1308 luamplib.preambles = {
1309   preamble = [[
1310 boolean mplib ; mplib := true ;
1311 let dump = endinput ;
1312 let normalfontsize = fontsize;
1313 input %s ;
1314 ]],
1315   mplibcode = [[
1316 texscriptmode := 2;
1317 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1318 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1319 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1320 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1321 if known context_mlib:
1322   defaultfont := "cmtt10";
1323   let infont = normalinfont;
1324   let fontsize = normalfontsize;
1325   vardef thelabel@#(expr p,z) =
1326     if string p :
1327       thelabel@#(p infont defaultfont scaled defaultscale,z)
1328     else :
1329       p shifted (z + labeloffset*mfun_laboff@# -
1330         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1331         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1332     fi
1333   enddef;
1334 else:
1335   vardef texttext@# primary t = rawtexttext (t) enddef;
1336   def message expr t =
1337     if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1338   enddef;
1339   def withtransparency (expr a, t) =
1340     withprescript "tr_alternative=" & if numeric a: decimal fi a
1341     withprescript "tr_transparency=" & decimal t
1342   enddef;
1343   vardef ddecimal primary p =
1344     decimal xpart p & " " & decimal ypart p
1345   enddef;
1346   vardef boundingbox primary p =
1347     if (path p) or (picture p) :
1348       llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1349     else :

```

```

1350     origin
1351   fi -- cycle
1352 enddef;
1353 fi
1354 def resolvedcolor(expr s) =
1355   runscript("return luamplib.shadecolor('"& s &"')")
1356 enddef;
1357 def colordecimals primary c =
1358   if cmykcolor c:
1359     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1360     decimal yellowpart c & ":" & decimal blackpart c
1361   elseif rgbcolor c:
1362     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1363   elseif string c:
1364     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1365   else:
1366     decimal c
1367   fi
1368 enddef;
1369 def externalfigure primary filename =
1370   draw rawtexttext("\includegraphics{"& filename &}")
1371 enddef;
1372 def TEX = texttext enddef;
1373 def mplibtexcolor primary c =
1374   runscript("return luamplib.gettexcolor('"& c &"')")
1375 enddef;
1376 def mplibrgbtexcolor primary c =
1377   runscript("return luamplib.gettexcolor('"& c &"', 'rgb')")
1378 enddef;
1379 def mplibgraphicstext primary t =
1380   begingroup;
1381   mplibgraphicstext_ (t)
1382 enddef;
1383 def mplibgraphicstext_ (expr t) text rest =
1384   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1385   fb, fc, dc, graphicstextpic, alsoordoublepath;
1386   picture graphicstextpic; graphicstextpic := nullpicture;
1387   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1388   let scale = scaled;
1389   def fakebold primary c = hide(fb:=c;) enddef;
1390   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1391   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1392   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1393   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1394   addto graphicstextpic alsoordoublepath (origin--cycle) rest; graphicstextpic:=nullpicture;
1395   def fakebold primary c = enddef;
1396   let fillcolor = fakebold; let drawcolor = fakebold;
1397   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1398   image(draw runscript("return luamplib.graphicstext([====["&t&"]====],")

```

```

1399   & decimal fb &"',"'& fc &"',"'& dc &"')") rest;)
1400 endgroup;
1401 enddef;
1402 def mplibglyph expr c of f =
1403   runscript (
1404     "return luamplib.glyph('"
1405     & if numeric f: decimal fi f
1406     & "'',"
1407     & if numeric c: decimal fi c
1408     & "')"
1409   )
1410 enddef;
1411 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1412 def mplibdrawglyph expr g =
1413   luamplib_tmp_num_ := 0;
1414   for item within g:
1415     fill pathpart item
1416     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1417   endfor
1418 enddef;
1419 let mplibfillglyph = mplibdrawglyph;
1420 def mplibstrokeglyph expr g =
1421   luamplib_tmp_num_ := 0;
1422   for item within g:
1423     draw pathpart item
1424     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1425   endfor
1426 enddef;
1427 def mplibfillandstrokeglyph expr g =
1428   luamplib_tmp_num_ := 0;
1429   for item within g:
1430     draw pathpart item withpostscript
1431     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1432   endfor
1433 enddef;
1434 def withmplibcolors (expr f, s) =
1435   runscript("return luamplib.fillandstrokecolor('" &
1436     if not string f: colordecimals fi f & "''," &
1437     if not string s: colordecimals fi s & "')"
1438 enddef;
1439 def withmplibopacities (expr a, f, s) =
1440   withprescript "tr_alternative=" & if numeric a: decimal fi a
1441   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1442 enddef;
1443 def mplib_do_outline_text_set_b (text f) (text d) text r =
1444   def mplib_do_outline_options_f = f enddef;
1445   def mplib_do_outline_options_d = d enddef;
1446   def mplib_do_outline_options_r = r enddef;
1447 enddef;

```

```

1448 def mplib_do_outline_text_set_f (text f) text r =
1449   def mplib_do_outline_options_f = f enddef;
1450   def mplib_do_outline_options_r = r enddef;
1451 enddef;
1452 def mplib_do_outline_text_set_u (text f) text r =
1453   def mplib_do_outline_options_f = f enddef;
1454 enddef;
1455 def mplib_do_outline_text_set_d (text d) text r =
1456   def mplib_do_outline_options_d = d enddef;
1457   def mplib_do_outline_options_r = r enddef;
1458 enddef;
1459 def mplib_do_outline_text_set_r (text d) (text f) text r =
1460   def mplib_do_outline_options_d = d enddef;
1461   def mplib_do_outline_options_f = f enddef;
1462   def mplib_do_outline_options_r = r enddef;
1463 enddef;
1464 def mplib_do_outline_text_set_n text r =
1465   def mplib_do_outline_options_r = r enddef;
1466 enddef;
1467 def mplib_do_outline_text_set_p = enddef;
1468 def mplib_fill_outline_text =
1469   for n=1 upto mpliboutlinenum:
1470     i:=0;
1471     for item within mpliboutlinepic[n]:
1472       i:=i+1;
1473       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1474       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1475     endfor
1476   endfor
1477 enddef;
1478 def mplib_draw_outline_text =
1479   for n=1 upto mpliboutlinenum:
1480     for item within mpliboutlinepic[n]:
1481       draw pathpart item mplib_do_outline_options_d;
1482     endfor
1483   endfor
1484 enddef;
1485 def mplib_filldraw_outline_text =
1486   for n=1 upto mpliboutlinenum:
1487     i:=0;
1488     for item within mpliboutlinepic[n]:
1489       i:=i+1;
1490       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1491         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1492       else:
1493         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1494       fi
1495     endfor
1496   endfor

```

```

1497 enddef;
1498 vardef mpliboutlinetext@# (expr t) text rest =
1499   save kind; string kind; kind := str @#;
1500   save i; numeric i;
1501   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1502   def mplib_do_outline_options_d = enddef;
1503   def mplib_do_outline_options_f = enddef;
1504   def mplib_do_outline_options_r = enddef;
1505   runscript("return luampplib.outlinetext[==["&t&"]==]");
1506   image ( addto currentpicture also image (
1507     if kind = "f":
1508       mplib_do_outline_text_set_f rest;
1509       mplib_fill_outline_text;
1510     elseif kind = "d":
1511       mplib_do_outline_text_set_d rest;
1512       mplib_draw_outline_text;
1513     elseif kind = "b":
1514       mplib_do_outline_text_set_b rest;
1515       mplib_fill_outline_text;
1516       mplib_draw_outline_text;
1517     elseif kind = "u":
1518       mplib_do_outline_text_set_u rest;
1519       mplib_filldraw_outline_text;
1520     elseif kind = "r":
1521       mplib_do_outline_text_set_r rest;
1522       mplib_draw_outline_text;
1523       mplib_fill_outline_text;
1524     elseif kind = "p":
1525       mplib_do_outline_text_set_p;
1526       mplib_draw_outline_text;
1527     else:
1528       mplib_do_outline_text_set_n rest;
1529       mplib_fill_outline_text;
1530     fi;
1531   ) mplib_do_outline_options_r; )
1532 enddef ;
1533 def withmppattern primary p =
1534   withprescript "mplibpattern=" & if numeric p: decimal fi p
1535 enddef;
1536 primarydef t withpattern p =
1537   image(
1538     if cycle t:
1539       fill
1540     else:
1541       draw
1542     fi
1543     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1544 enddef;
1545 vardef mplibtransformmatrix (text e) =

```

```

1546 save t; transform t;
1547 t = identity e;
1548 runscript("luamplib.transformmatrix = {"
1549 & decimal xpart t & ","
1550 & decimal ypart t & ","
1551 & decimal xpart t & ","
1552 & decimal ypart t & ","
1553 & decimal xpart t & ","
1554 & decimal ypart t & ","
1555 & "}");
1556 enddef;
1557 primarydef p withmaskinggroup s =
1558   if picture p:
1559     image(
1560       draw p;
1561       draw center p withprescript "mplibfadestate=stop";
1562     )
1563   else:
1564     p withprescript "mplibfadestate=stop"
1565   fi
1566   withprescript "mplibfadetype=masking"
1567   withprescript "mplibmaskname=" & s
1568 enddef;
1569 def withmaskingbgcolor expr c =
1570   withprescript "mplibmaskingbgcolor=" & colordecimals c
1571 enddef;
1572 primarydef p withfademethod s =
1573   if picture p:
1574     image(
1575       draw p;
1576       draw center p withprescript "mplibfadestate=stop";
1577     )
1578   else:
1579     p withprescript "mplibfadestate=stop"
1580   fi
1581   withprescript "mplibfadetype=" & s
1582   withprescript "mplibfadebbox=" &
1583     decimal (xpart llcorner p -1/4) & ":" &
1584     decimal (ypart llcorner p -1/4) & ":" &
1585     decimal (xpart urcorner p +1/4) & ":" &
1586     decimal (ypart urcorner p +1/4)
1587 enddef;
1588 def withfadeopacity (expr a,b) =
1589   withprescript "mplibfadeopacity=" &
1590     decimal a & ":" &
1591     decimal b
1592 enddef;
1593 def withfadevector (expr a,b) =
1594   withprescript "mplibfadevector=" &

```

```

1595     decimal xpart a & ":" &
1596     decimal ypart a & ":" &
1597     decimal xpart b & ":" &
1598     decimal ypart b
1599 enddef;
1600 let withfadecenter = withfadevector;
1601 def withfaderadius (expr a,b) =
1602   withprescript "mplibfaderadius=" &
1603     decimal a & ":" &
1604     decimal b
1605 enddef;
1606 def withfadebbox (expr a,b) =
1607   withprescript "mplibfadebbox=" &
1608     decimal xpart a & ":" &
1609     decimal ypart a & ":" &
1610     decimal xpart b & ":" &
1611     decimal ypart b
1612 enddef;
1613 primarydef p asgroup s =
1614   image(
1615     draw center p
1616       withprescript "mplibgroupbbox=" &
1617         decimal (xpart llcorner p -1/4) & ":" &
1618         decimal (ypart llcorner p -1/4) & ":" &
1619         decimal (xpart urcorner p +1/4) & ":" &
1620         decimal (ypart urcorner p +1/4)
1621       withprescript "gr_state=start"
1622       withprescript "gr_type=" & s;
1623     draw p withprescript "sh_in_xobj=yes";
1624     draw center p withprescript "gr_state=stop";
1625   )
1626 enddef;
1627 def withgroupbbox (expr a,b) =
1628   withprescript "mplibgroupbbox=" &
1629     decimal xpart a & ":" &
1630     decimal ypart a & ":" &
1631     decimal xpart b & ":" &
1632     decimal ypart b
1633 enddef;
1634 def withgroupname expr s =
1635   withprescript "mplibgroupname=" & s
1636 enddef;
1637 def usemplibgroup primary s =
1638   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s &"\endcsname}")
1639   shifted runscript("return luamplib.trgroupshifts["' & s & "']")
1640 enddef;
1641 path    mplib_shade_path ;
1642 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1643 numeric mplib_shade_fx, mplib_shade_fy ;

```



```

1644 numeric mplib_shade_lx, mplib_shade_ly ;
1645 numeric mplib_shade_nx, mplib_shade_ny ;
1646 numeric mplib_shade_dx, mplib_shade_dy ;
1647 numeric mplib_shade_tx, mplib_shade_ty ;
1648 primarydef p withshadingmethod m =
1649   p
1650   if picture p :
1651     withprescript "sh_operand_type=picture"
1652     if textual p or (length p > 1):
1653       withprescript "sh_transform=no"
1654       mplib_with_shade_method (boundingbox p, m)
1655     else:
1656       withprescript "sh_transform=yes"
1657       mplib_with_shade_method (pathpart p, m)
1658     fi
1659   else :
1660     withprescript "sh_transform=yes"
1661     mplib_with_shade_method (p, m)
1662   fi
1663 enddef;
1664 def mplib_with_shade_method (expr p, m) =
1665   hide(mplib_with_shade_method_analyze(p))
1666   withprescript "sh_domain=0 1"
1667   withprescript "sh_color=into"
1668   withprescript "sh_color_a=" & colordecimals white
1669   withprescript "sh_color_b=" & colordecimals black
1670   withprescript "sh_first=" & ddecimal point 0 of p
1671   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1672   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1673   if m = "linear" :
1674     withprescript "sh_type=linear"
1675     withprescript "sh_factor=1"
1676     withprescript "sh_center_a=" & ddecimal llcorner p
1677     withprescript "sh_center_b=" & ddecimal urcorner p
1678   else :
1679     withprescript "sh_type=circular"
1680     withprescript "sh_factor=1.2"
1681     withprescript "sh_center_a=" & ddecimal center p
1682     withprescript "sh_center_b=" & ddecimal center p
1683     withprescript "sh_radius_a=" & decimal 0
1684     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1685   fi
1686 enddef;
1687 def mplib_with_shade_method_analyze(expr p) =
1688   mplib_shade_path := p ;
1689   mplib_shade_step := 1 ;
1690   mplib_shade_fx := xpart point 0 of p ;
1691   mplib_shade_fy := ypart point 0 of p ;
1692   mplib_shade_lx := mplib_shade_fx ;

```

```

1693 mplib_shade_ly := mplib_shade_fy ;
1694 mplib_shade_nx := 0 ;
1695 mplib_shade_ny := 0 ;
1696 mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1697 mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1698 for i=1 upto length(p) :
1699     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1700     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1701     if mplib_shade_tx > mplib_shade_dx :
1702         mplib_shade_nx := i + 1 ;
1703         mplib_shade_lx := xpart point i of p ;
1704         mplib_shade_dx := mplib_shade_tx ;
1705     fi ;
1706     if mplib_shade_ty > mplib_shade_dy :
1707         mplib_shade_ny := i + 1 ;
1708         mplib_shade_ly := ypart point i of p ;
1709         mplib_shade_dy := mplib_shade_ty ;
1710     fi ;
1711 endfor ;
1712 enddef;
1713 vardef mplib_max_radius(expr p) =
1714     max (
1715         (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1716         (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1717         (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1718         (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1719     )
1720 enddef;
1721 def withshadingstep (text t) =
1722     hide(mplib_shade_step := mplib_shade_step + 1 ;)
1723     withprescript "sh_step=" & decimal mplib_shade_step
1724     t
1725 enddef;
1726 def withshadingradius expr a =
1727     withprescript "sh_radius_a=" & decimal (xpart a)
1728     withprescript "sh_radius_b=" & decimal (ypart a)
1729 enddef;
1730 def withshadingorigin expr a =
1731     withprescript "sh_center_a=" & ddecimal a
1732     withprescript "sh_center_b=" & ddecimal a
1733 enddef;
1734 def withshadingvector expr a =
1735     withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1736     withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1737 enddef;
1738 def withshadingdirection expr a =
1739     withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1740     withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1741 enddef;

```

```

1742 def withshadingtransform expr a =
1743   withprescript "sh_transform=" & a
1744 enddef;
1745 def withshadingcenter expr a =
1746   withprescript "sh_center_a=" & ddecimal (
1747     center mplib_shade_path shifted (
1748       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1749       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1750     )
1751   )
1752 enddef;
1753 def withshadingcenters (expr a, b) =
1754   withprescript "sh_center_a=" & ddecimal a
1755   withprescript "sh_center_b=" & ddecimal b
1756   withshadingtransform "no"
1757   withshadingfactor 1
1758 enddef;
1759 let withshadingpoints = withshadingcenters;
1760 def withshadingextend (expr a, b) =
1761   withprescript "sh_extend=" &
1762   if a: "true" else: "false" fi & " " &
1763   if b: "true" else: "false" fi
1764 enddef;
1765 def withshadingdomain expr d =
1766   withprescript "sh_domain=" & ddecimal d
1767 enddef;
1768 def withshadingfactor expr f =
1769   withprescript "sh_factor=" & decimal f
1770 enddef;
1771 def withshadingfraction expr a =
1772   if mplib_shade_step > 0 :
1773     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1774   fi
1775 enddef;
1776 def withshadingcolors (expr a, b) =
1777   if mplib_shade_step > 0 :
1778     withprescript "sh_color=into"
1779     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1780     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1781   else :
1782     withprescript "sh_color=into"
1783     withprescript "sh_color_a=" & colordecimals a
1784     withprescript "sh_color_b=" & colordecimals b
1785   fi
1786 enddef;
1787 def withshadingstroke expr a =
1788   withprescript "sh_stroking=" & a
1789 enddef;
1790 def mpliblength primary t =

```

```

1791 runscript("return utf8.len[==[" & t & "]==]")
1792 enddef;
1793 def mplibsubstring expr p of t =
1794   runscript("return luamplib.unicodesubstring([==[" & t & "]==],"
1795     & decimal xpart p & ","
1796     & decimal ypart p & ")")
1797 enddef;
1798 def mlibuclength primary t =
1799   runscript("return #luamplib.getunicodegraphemes([==[" & t & "]==]")
1800 enddef;
1801 def mlibucsubstring expr p of t =
1802   runscript("return luamplib.unicodesubstring([==[" & t & "]==],"
1803     & decimal xpart p & ","
1804     & decimal ypart p & ",true)")
1805 enddef;
1806 ]],
1807 legacyverbatimtex = [[
1808 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1809 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1810 let VerbatimTeX = specialVerbatimTeX;
1811 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1812   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1813 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1814   "runscript(" &ditto&
1815   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1816   "luamplib.in_the_fig=false" &ditto& ");";
1817 ]],
1818 texttextlabel = [[
1819 let luampliboriginalinfont = infont;
1820 primarydef s infont f =
1821   if (s < char 32)
1822     or (s = char 35) % #
1823     or (s = char 36) % $
1824     or (s = char 37) % %
1825     or (s = char 38) % &
1826     or (s = char 92) % \
1827     or (s = char 94) % ^
1828     or (s = char 95) % _
1829     or (s = char 123) % {
1830     or (s = char 125) % }
1831     or (s = char 126) % ~
1832     or (s = char 127) :
1833     s luampliboriginalinfont f
1834   else :
1835     rawtexttext(s)
1836   fi
1837 enddef;
1838 def fontsize expr f =
1839   begingroup

```

```

1840 save size; numeric size;
1841 size := mplibdimen("1em");
1842 if size = 0: 10pt else: size fi
1843 endgroup
1844 enddef;
1845 ]],
1846 }
1847

```

process_mplibcode

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

```

1848 luamplib.verbatiminput = false
1849 luamplib.everymplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1850 luamplib.everyendmplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1851 function luamplib.process_mplibcode (data, instancename)
1852 texboxes.localid = 4096

```

This is needed for legacy behavior

```

1853 if luamplib.legacyverbatim then
1854   luamplib.figid, tex_code_pre_mplib = 1, {}
1855 end
1856 local everymplib = luamplib.everymplib[instancename]
1857 local everyendmplib = luamplib.everyendmplib[instancename]
1858 data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
1859 :gsub("\r","\n")

```

These five lines are needed for mplibverbatim mode.

```

1860 if luamplib.verbatiminput then
1861   data = data:gsub("\mpcolor%s+(-%b{ })", "mplibcolor(\"%1\")")
1862   :gsub("\mpdim%s+(-%b{ })", "mplibdimen(\"%1\")")
1863   :gsub("\mpdim%s+(-%a+)", "mplibdimen(\"%1\")")
1864   :gsub(btex_etex, "btex %1 etex ")
1865   :gsub(verbatimtex_etex, "verbatimtex %1 etex;")
1866 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, verbatimtex ... etex, and string expressions.

```

1867   local t = { } -- to store btex, verbatimtex, string
1868   data = data:gsub(btex_etex, function(str)
1869     t[#t+1] = str
1870     return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
1871   end)
1872   :gsub(verbatimtex_etex, function(str)
1873     t[#t+1] = str
1874     return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
1875   end)
1876   :gsub('\"(.-)\"', function(str)
1877     t[#t+1] = str
1878     return format('\"\\unexpanded{!l!u!a!%s!m!p!l!}\"', #t)

```

```

1879 end)
1880 :gsub("\\%", "\\0PerCent\\0")
1881 :gsub("%%.-\\n", "\\n")
1882 :gsub("%zPerCent%z", "\\%")
1883 run_tex_code(format("\\mplibtmptoks\\expandafter{\\expanded{%s}}",data))
1884 data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

1885 :gsub("##", "#")
1886 :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
1887 end
1888 process(data, instancename)
1889 end
1890

```

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.

```

1891 local figcontents = { post = { } }
1892 local function put2output(a,...)
1893 figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
1894 end
1895 local function pdf_startfigure(n,llx,lly,urx,ury)
1896 put2output("\\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
1897 end
1898 local function pdf_stopfigure()
1899 put2output("\\mplibstoptoPDF")
1900 end

```

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

```

1901 local function pdf_literalcode (...)
1902 put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
1903 end
1904 local start_pdf_code = pdfmode
1905 and function() pdf_literalcode"q" end
1906 or function() put2output"\\special{pdf:bcontent}" end
1907 local stop_pdf_code = pdfmode
1908 and function() pdf_literalcode"Q" end
1909 or function() put2output"\\special{pdf:econtent}" end
1910

```

Now we process hboxes created from btex ... etex or texttext(...) or TEX(...) etc.

```

1911 local function put_tex_boxes (object,prescript)
1912 local box = prescript.mplibtexboxid:explode":"
1913 local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
1914 if n and tw and th then
1915 local op = object.path
1916 local first, second, fourth = op[1], op[2], op[4]
1917 local tx, ty = first.x_coord, first.y_coord
1918 local sx, rx, ry, sy = 1, 0, 0, 1
1919 if tw ~= 0 then
1920 sx = (second.x_coord - tx)/tw

```

```

1921     rx = (second.y_coord - ty)/tw
1922     if sx == 0 then sx = 0.00001 end
1923 end
1924 if th ~= 0 then
1925     sy = (fourth.y_coord - ty)/th
1926     ry = (fourth.x_coord - tx)/th
1927     if sy == 0 then sy = 0.00001 end
1928 end
1929 start_pdf_code()
1930 pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
1931 put2output("\\mplibputtextbox{%i}",n)
1932 stop_pdf_code()
1933 end
1934 end
1935

```

Colors

```

1936 local do_preobj_CR
1937 do
1938   local prev_override_color
1939   function do_preobj_CR(object,prescript)
1940     if object.postscript == "collect" then return end
1941     local override = prescript and prescript.mpliboverridecolor
1942     if override then
1943       if pdfmode then
1944         pdf_literalcode(override)
1945         override = nil
1946       else
1947         put2output("\\special{%s}",override)
1948         prev_override_color = override
1949       end
1950     else
1951       local cs = object.color
1952       if cs and #cs > 0 then
1953         pdf_literalcode(luamplib.colorconverter(cs))
1954         prev_override_color = nil
1955       elseif not pdfmode then
1956         override = prev_override_color
1957         if override then
1958           put2output("\\special{%s}",override)
1959         end
1960       end
1961     end
1962     return override
1963   end
1964 end
1965

```

For transparency, shading, fading, and pattern

```

1966 local pdfmanagement = is_defined'pdfmanagement_add:nnn'

```

```

1967 local pdfobjs, pdfetcs = {}, {}
1968 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
1969 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
1970 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
1971 local function update_pdfobjs (os, stream)
1972   local key = os
1973   if stream then key = key..stream end
1974   local on = key and pdfobjs[key]
1975   if on then
1976     return on,false
1977   end
1978   if pdfmode then
1979     if stream then
1980       on = pdf.immediateobj("stream",stream,os)
1981     elseif os then
1982       on = pdf.immediateobj(os)
1983     else
1984       on = pdf.reserveobj()
1985     end
1986   else
1987     on = pdfetcs.cnt or 1
1988     if stream then
1989       texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
1990     elseif os then
1991       texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
1992     else
1993       texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
1994     end
1995     pdfetcs.cnt = on + 1
1996   end
1997   if key then
1998     pdfobjs[key] = on
1999   end
2000   return on,true
2001 end
2002 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2003 if pdfmode then
2004   pdfetcs.getpageres = pdf.getpageresources or function() return pdf.pageresources end
2005   local getpageres = pdfetcs.getpageres
2006   local setpageres = pdf.setpageresources or function(s) pdf.pageresources = s end
2007   local initialize_resources = function (name)
2008     local tabname = format("%s_res",name)
2009     pdfetcs[tabname] = { }
2010     if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
2011       local obj = pdf.reserveobj()
2012       setpageres(format("%s/%s %i 0 R", getpageres() or "", name, obj))
2013       luatexbase.add_to_callback("finish_pdffile", function()
2014         pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
2015       end,

```



```

2016     format("luamplib.%s.finish_pdffile",name))
2017 end
2018 end
2019 pdfetcs.fallback_update_resources = function (name, res)
2020     local tabname = format("%s_res",name)
2021     if not pdfetcs[tabname] then
2022         initialize_resources(name)
2023     end
2024     if luatexbase.callbacktypes.finish_pdffile then
2025         local t = pdfetcs[tabname]
2026         t[#t+1] = res
2027     else
2028         local tpr, n = getpagers() or "", 0
2029         tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2030         if n == 0 then
2031             tpr = format("%s/%s<<%s>>", tpr, name, res)
2032         end
2033         setpagers(tpr)
2034     end
2035 end
2036 else
2037     texsprint {
2038         "\\luamplibatfirstshipout{",
2039         "\\special{pdf:obj @MPlibTr<<>>}",
2040         "\\special{pdf:obj @MPlibSh<<>>}",
2041         "\\special{pdf:obj @MPlibCS<<>>}",
2042         "\\special{pdf:obj @MPlibPt<<>>}}",
2043     }
2044     pdfetcs.fallback_update_resources = function (name,res,obj)
2045         texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2046         local tabname = format("%s_res",name)
2047         if not pdfetcs[tabname] then
2048             texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2049             pdfetcs[tabname] = { }
2050         end
2051         tableinsert(pdfetcs[tabname], res)
2052     end
2053 end
2054

```

Transparency

```

2055 local function add_extgs_resources (on, new)
2056     local key = format("MPlibTr%s", on)
2057     if new then
2058         local val = format(pdfetcs.resfmt, on)
2059         if pdfmanagement then
2060             texsprint {
2061                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{", val, "}"
2062             }

```

```

2063     else
2064         local tr = format("/%s %s", key, val)
2065         if is_defined(pdfetcs.pgfbx) then
2066             texsprintf { "\\csname ", pdfetcs.pgfbx, "\\endcsname{", tr, "}" }
2067         elseif is_defined"TRP@list" then
2068             texsprintf(catat11,{
2069                 [[\if@files\immediate\write\@auxout{]],
2070                 [[\string\g@addto@macro\string\TRP@list{]],
2071                 tr,
2072                 [[}]\fi]],
2073             })
2074             if not get_macro"TRP@list":find(tr) then
2075                 texsprintf(catat11,[[\global\TRP@runtrue]])
2076             end
2077         else
2078             pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2079         end
2080     end
2081 end
2082 return key
2083 end
2084
2085 local do_preobj_TR
2086 do
2087     local transparency_modes = {
2088         [0] = "Normal",
2089         "Normal",      "Multiply",      "Screen",      "Overlay",
2090         "SoftLight",    "HardLight",    "ColorDodge",  "ColorBurn",
2091         "Darken",       "Lighten",    "Difference",  "Exclusion",
2092         "Hue",          "Saturation", "Color",       "Luminosity",
2093         "Compatible",
2094         normal         = "Normal",      multiply = "Multiply",  screen   = "Screen",
2095         overlay        = "Overlay",      softlight = "SoftLight", hardlight = "HardLight",
2096         colorDodge     = "ColorDodge",    colorburn = "ColorBurn", darken    = "Darken",
2097         lighten        = "Lighten",       difference = "Difference", exclusion = "Exclusion",
2098         hue            = "Hue",           saturation = "Saturation", color     = "Color",
2099         luminosity     = "Luminosity",    compatible = "Compatible",
2100     }
2101     function do_preobj_TR(object,prescript)
2102         if object.postscript == "collect" then return end
2103         local opaq = prescript and prescript.tr_transparency
2104         if not opaq then return end
2105
2106         local key, on, os, new
2107         local mode = prescript.tr_alternative or 1
2108         mode = transparency_modes[tonumber(mode) or mode:lower()]
2109         if not mode then
2110             mode = prescript.tr_alternative
2111             warn("unsupported blend mode: '%s'", mode)

```

```

2112     end
2113     opaq = opaq:explode":""
2114     for i,v in ipairs(opaq) do
2115         opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2116     end
2117     for i,v in ipairs[ {mode,opaq[1],opaq[2] or opaq[1]},{"Normal",1,1} ] do
2118         os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2119         on, new = update_pdfobjs(os)
2120         key = add_extgs_resources(on,new)
2121         if i == 1 then
2122             pdf_literalcode("/%s gs",key)
2123         else
2124             return format("/%s gs",key)
2125         end
2126     end
2127 end
2128 end
2129

```

Shading with *metafun* format.

```

2130 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2131     for _,v in ipairs{ca,cb} do
2132         for i,vv in ipairs(v) do
2133             for ii,vvv in ipairs(vv) do
2134                 v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2135             end
2136         end
2137     end
2138     local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2139     if steps > 1 then
2140         local list,bounds,encode = { },{ },{ }
2141         for i=1,steps do
2142             if i < steps then
2143                 bounds[i] = format("%.3f", fractions[i] or 1)
2144             end
2145             encode[2*i-1] = 0
2146             encode[2*i] = 1
2147             os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
2148             :gsub(decimals,rmzeros)
2149             list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2150         end
2151         os = tableconcat {
2152             "<</FunctionType 3",
2153             format("/Bounds[%s]", tableconcat(bounds, ' ')),
2154             format("/Encode[%s]", tableconcat(encode, ' ')),
2155             format("/Functions[%s]", tableconcat(list, ' ')),
2156             format("/Domain[%s]>>", domain),
2157         } :gsub(decimals,rmzeros)
2158     else

```

```

2159     os = fun2fmt:format(domain,tableconcat(ca[1],' '),tableconcat(cb[1],' '))
2160     :gsub(decimals,rmzeros)
2161 end
2162 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2163 os = tableconcat {
2164     format("</ShadingType %i", shtype),
2165     format("/ColorSpace %s",    colorspace),
2166     format("/Function %s",      objref),
2167     format("/Coords[%s]",      coordinates),
2168     format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2169 } :gsub(decimals,rmzeros)
2170 local on, new = update_pdfobjs(os)
2171 if new then
2172     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2173     if pdfmanagement then
2174         texsprint {
2175             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2176         }
2177     else
2178         local res = format("/%s %s", key, val)
2179         pdfetcs.fallback_update_resources("Shading",res,"@MPlibSh")
2180     end
2181 end
2182 return on
2183 end
2184
2185 local do_preobj_SH
2186 do
2187     pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2188         run_tex_code({
2189             [[\color_model_new:nnn]],
2190             format("{mplibcolorspace_%s}", names:gsub(",","_")),
2191             format("{DeviceN}{names={%s}}", names),
2192             [[\edef\mplib_atempa{\pdf_object_ref_last:}]],
2193         }, ccexplat)
2194         local colorspace = get_macro'mplib_atempa'
2195         t[names] = colorspace
2196         return colorspace
2197     end })
2198     local function color_normalize(ca,cb)
2199         if #cb == 1 then
2200             if #ca == 4 then
2201                 cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2202             else -- #ca = 3
2203                 cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2204             end
2205         elseif #cb == 3 then -- #ca == 4
2206             cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
2207         end

```

```

2208 end
2209 function do_preobj_SH(object, prescript)
2210     local shade_no
2211     local sh_type = prescript and prescript.sh_type
2212     if not sh_type then return end
2213
2214     local domain = prescript.sh_domain or "0 1"
2215     local centera = (prescript.sh_center_a or "0 0"):explode()
2216     local centerb = (prescript.sh_center_b or "0 0"):explode()
2217     local transform = prescript.sh_transform == "yes"
2218     local sx,sy,sr,dx,dy = 1,1,1,0,0
2219     if transform then
2220         local first = (prescript.sh_first or "0 0"):explode()
2221         local setx = (prescript.sh_set_x or "0 0"):explode()
2222         local sety = (prescript.sh_set_y or "0 0"):explode()
2223         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2224         if x ~= 0 and y ~= 0 then
2225             local path = object.path
2226             local path1x = path[1].x_coord
2227             local path1y = path[1].y_coord
2228             local path2x = path[x].x_coord
2229             local path2y = path[y].y_coord
2230             local dxa = path2x - path1x
2231             local dya = path2y - path1y
2232             local dxb = setx[2] - first[1]
2233             local dyb = sety[2] - first[2]
2234             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2235                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
2236                 sy = dya / dyb ; if sy < 0 then sy = - sy end
2237                 sr = math.sqrt(sx^2 + sy^2)
2238                 dx = path1x - sx*first[1]
2239                 dy = path1y - sy*first[2]
2240             end
2241         end
2242     end
2243     local ca, cb, colorspace, steps, fractions
2244     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:" }
2245     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:" }
2246     steps = tonumber(prescript.sh_step) or 1
2247     if steps > 1 then
2248         fractions = { prescript.sh_fraction_1 or 0 }
2249         for i=2,steps do
2250             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2251             ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
2252             cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
2253         end
2254     end
2255     if prescript.mplib_spotcolor then
2256         ca, cb = { }, { }

```

```

2257 local names, pos, objref = { }, -1, ""
2258 local script = object.prescript:explode"\13+"
2259 for i=#script,1,-1 do
2260     if script[i]:find"mplib_spotcolor" then
2261         local t, name, value = script[i]:explode"="[2]:explode":"
2262         value, objref, name = t[1], t[2], t[3]
2263         if not names[name] then
2264             pos = pos+1
2265             names[name] = pos
2266             names[#names+1] = name
2267         end
2268         t = { }
2269         for j=1,names[name] do t[#t+1] = 0 end
2270         t[#t+1] = value
2271         tableinsert(#ca == #cb and ca or cb, t)
2272     end
2273 end
2274 for _,t in ipairs{ca,cb} do
2275     for _,tt in ipairs(t) do
2276         for i=1,#names-#tt do tt[#tt+1] = 0 end
2277     end
2278 end
2279 if #names == 1 then
2280     colorspace = objref
2281 else
2282     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2283 end
2284 else
2285     local model = 0
2286     for _,t in ipairs{ca,cb} do
2287         for _,tt in ipairs(t) do
2288             model = model > #tt and model or #tt
2289         end
2290     end
2291     for _,t in ipairs{ca,cb} do
2292         for _,tt in ipairs(t) do
2293             if #tt < model then
2294                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2295             end
2296         end
2297     end
2298     colorspace = model == 4 and "/DeviceCMYK"
2299                 or model == 3 and "/DeviceRGB"
2300                 or model == 1 and "/DeviceGray"
2301                 or err"unknown color model"
2302 end
2303 local extend = prescript.sh_extend
2304 if sh_type == "linear" then
2305     local coordinates = format("%f %f %f %f",

```

```

2306     dx + sx*centera[1], dy + sy*centera[2],
2307     dx + sx*centerb[1], dy + sy*centerb[2])
2308     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2309 elseif sh_type == "circular" then
2310     local factor = prescript.sh_factor or 1
2311     local radiusa = factor * prescript.sh_radius_a
2312     local radiusb = factor * prescript.sh_radius_b
2313     local coordinates = format("%f %f %f %f %f %f",
2314         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2315         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2316     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2317 else
2318     err"unknown shading type"
2319 end
2320 return shade_no, prescript.sh_stroking == "yes"
2321 end
2322 end
2323

```

Shading Patterns: we can apply shading to textual pictures as well as paths.

```

2324 local function add_pattern_resources (key, val)
2325   if pdfmanagement then
2326     texsprint {
2327       "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2328     }
2329   else
2330     local res = format("/%s %s", key, val)
2331     if is_defined(pdfetcs.pgfpattern) then
2332       texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2333     else
2334       pdfetcs.fallback_update_resources("Pattern",res,"@MPlibPt")
2335     end
2336   end
2337 end
2338 if not pdfmode then
2339   pdfetcs.shadingpatterns = { }
2340   pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2341 end
2342 function luamplib.dolatelua (on, os, xobj)
2343   local h, v = pdf.getpos()
2344   h = format("%f", h/factor) :gsub(decimals,rmzeros)
2345   v = format("%f", v/factor) :gsub(decimals,rmzeros)
2346   if pdfmode then
2347     pdf.obj(on, format("<<%s/Matrix[1 0 0 1 %s %s]>>", os, h, v))
2348     pdf.refobj(on)
2349   else
2350     local t = pdfetcs.shadingpatterns[on] or { }
2351     local shift = os == "group" and pdfetcs.tr_group.shifts[xobj]
2352       or os == "pattern" and pdfetcs.patterns[xobj].shifts

```

```

2353   if shift then
2354       h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2355   end
2356   if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2357       warn"Rerun to get correct shading pattern"
2358   end
2359   local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2360   local init = pdfetcs.shadingpatterninit_w
2361   if init then pdfetcs.shadingpatterninit_w = nil end
2362   local f = ioopen(name, init and "w" or "a")
2363   if f then
2364       f:write(("s %s %s\n"):format(on, h, v))
2365       f:close()
2366   else
2367       err"cannot write a file. check the cache dir path"
2368   end
2369 end
2370 end
2371 local function do_preobj_shading (object, prescript)
2372   if not prescript or not prescript.sh_operand_type then return end
2373   local on = do_preobj_SH(object, prescript)
2374   local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2375   if prescript.sh_in_xobj == "yes" then
2376       on = update_pdfobjs(("<<%s>>"):format(os))
2377       goto skip_latelua
2378   end
2379   on = update_pdfobjs()
2380   if pdfmode then
2381       put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(",on,",[",os,"]] }" })
2382   else
2383       local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2384       or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2385       if xobj or not is_defined"RecordProperties" then -- in xobject or plain
2386           local init = pdfetcs.shadingpatterninit_r
2387           if init then
2388               pdfetcs.shadingpatterninit_r = nil
2389               local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2390               local f = ioopen(name)
2391               if f then
2392                   for line in f:lines() do
2393                       local t = line:explode()
2394                       pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2395                   end
2396                   f:close()
2397               end
2398           end
2399           local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2400           texsprint{ "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2401               format(" <<%s/Matrix[1 0 0 1 %s %s]>>", os, t[1], t[2]) }

```



```

2441 elseif is_defined(pdfetcs.pgfbextgs) then
2442   run_tex_code"\relax" -- flush tex.sprint queue
2443   if pdfmode then
2444     for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2445                       ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2446                       Pattern     = "pgf@sys@pgf@resource@list@patterns", } do
2447       local res = (get_macro(v) or ""):gsub("^%s*(.)%s*$", "%1")
2448       if res ~= "" then
2449         t[#t+1] = ("%s<<%s>>"):format(k, res )
2450       end
2451     end
2452   else
2453     local abc = get_macro"pgfutil@abc" or ""
2454     for k,v in pairs { ExtGState = "@pgfbextgs",
2455                       ColorSpace = "@pgfcolorspaces",
2456                       Pattern     = "@pgfpatterns", } do
2457       local tt = { }
2458       for vv in abc:gmatch( v .. "%s*(%b<>)" ) do
2459         tt[#tt+1] = vv:match("^<<%s*(.)%s*>>$")
2460       end
2461       if #tt > 0 then
2462         t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2463       end
2464     end
2465   end

```

We still have to deal with Shading resources.

```

2466   if luatexbase.callbacktypes.finish_pdffile then
2467     if pdfetcs.Shading_res then
2468       t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2469     end
2470   else
2471     local res = pdfetcs.getpageres()
2472     res = res and res:match"/Shading%s*%b<>"
2473     if res then
2474       t[#t+1] = res
2475     end
2476   end
2477 else
2478   if ispattern and is_defined"TRP@list" then

```

We do not gather transparent package's \TRP@list as Acrobat glitches on tiling pattern plus masking group, so warn users and recommend \DocumentMetadata

```

2479     warn"transparent package is not fully functional without pdfmanagement code."
2480   end
2481   if luatexbase.callbacktypes.finish_pdffile then
2482     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2483       local tt = pdfetcs[v.."_res"]
2484       if tt then
2485         t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))

```

```

2486     end
2487   end
2488   else
2489     local res = pdfetcs.getpageres()
2490     if res then
2491       t[#t+1] = res
2492     end
2493   end
2494 end
2495 local result = tableconcat(t)
2496 if optres ~= "" then
2497   for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2498     local res = optres:match("/"..v.."%"s*%b<>")
2499     if res then
2500       if result:find("/"..v) then
2501         res = res:match("<<(.+)>>$")
2502         result = result:gsub("/"..v.."%"s*<<," %1"..res, 1)
2503       else
2504         result = result .. res
2505       end
2506     end
2507   end
2508 end
2509 return result
2510 end
2511 function luamplib.registerpattern ( boxid, name, opts )
2512   local box = texgetbox(boxid)
2513   local wd = format("%.3f",box.width/factor)
2514   local hd = format("%.3f", (box.height+box.depth)/factor)
2515   info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2516   if opts.xstep == 0 then opts.xstep = nil end
2517   if opts.ystep == 0 then opts.ystep = nil end
2518   if opts.colored == nil then
2519     opts.colored = opts.coloured
2520     if opts.colored == nil then
2521       opts.colored = true
2522     end
2523   end
2524   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2525   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2526   if opts.matrix and opts.matrix:find"%a" then
2527     local data = format("mplibtransformmatrix(%s);",opts.matrix)
2528     process(data,"@mplibtransformmatrix")
2529     local t = luamplib.transformmatrix
2530     opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2531     opts.xshift = opts.xshift or format("%f",t[5])
2532     opts.yshift = opts.yshift or format("%f",t[6])
2533   end
2534   local attr = {

```

```

2535     "/Type/Pattern",
2536     "/PatternType 1",
2537     format("/PaintType %i", opts.colored and 1 or 2),
2538     "/TilingType 2",
2539     format("/XStep %s", opts.xstep or wd),
2540     format("/YStep %s", opts.ystep or hd),
2541     format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2542 }
2543 local optres = opts.resources or ""
2544 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2545 local patterns = pdfetcs.patterns
2546 if pdfmode then
2547     if opts.bbox then
2548         attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2549     end
2550     attr = tableconcat(attr) :gsub(decimals,rmzeros)
2551     local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2552     patterns[name] = { id = index, colored = opts.colored }
2553 else
2554     local cnt = #patterns + 1
2555     local objname = "@mplibpattern" .. cnt
2556     local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2557     texsprint {
2558         "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2559         "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2560         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2561         "\\special{pdf:bcontent}",
2562         "\\special{pdf:bxobj ", objname, " ", metric, "}",
2563         "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2564         "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2565         "\\special{pdf:put @resources <<", optres, ">>}",
2566         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2567         "\\special{pdf:econtent}}",
2568     }
2569     patterns[cnt] = objname
2570     patterns[name] = { id = cnt, colored = opts.colored }
2571     patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
2572 end
2573 end
2574
2575 local do_preobj_PAT
2576 do
2577     local function pattern_colorspace (cs)
2578         local on, new = update_pdfobjs(format("/Pattern %s]", cs))
2579         if new then
2580             local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
2581             if pdfmanagement then
2582                 texsprint {
2583                     "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"

```

```

2584     }
2585   else
2586     local res = format("/%s %s", key, val)
2587     if is_defined(pdfetcs.pgfcolorspace) then
2588       texsprintf { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
2589     else
2590       pdfetcs.fallback_update_resources("ColorSpace",res,"@MPLibCS")
2591     end
2592   end
2593 end
2594 return on
2595 end
2596 function do_preobj_PAT(object, prescript)
2597   local name = prescript and prescript.mplibpattern
2598   if not name then return end
2599   local patterns = pdfetcs.patterns
2600   local patt = patterns[name]
2601   local index = patt and patt.id or err("cannot get pattern object '%s'", name)
2602   local key = format("MPLibPt%s",index)
2603   if patt.colored then
2604     pdf_literalcode("/Pattern cs /%s scn", key)
2605   else
2606     local color = prescript.mpliboverridecolor
2607     if not color then
2608       local t = object.color
2609       color = t and #t>0 and luamplib.colorconverter(t)
2610     end
2611     if not color then return end
2612     local cs
2613     if color:find" cs " or color:find"@pdf.obj" then
2614       local t = color:explode()
2615       if pdfmode then
2616         cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
2617         color = t[3]
2618       else
2619         cs = t[2]
2620         color = t[3]:match"%[(.+)%"
2621       end
2622     else
2623       local t = colorsplit(color)
2624       cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
2625       color = tableconcat(t," ")
2626     end
2627     pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
2628   end
2629   if not patt.done then
2630     local val = pdfmode and format("%s 0 R",index) or patterns[index]
2631     add_pattern_resources(key,val)
2632   end

```

```

2633     patt.done = true
2634 end
2635 end
2636

```

Fading

```

2637 pdfetcs.fading = { }
2638 local function do_preobj_FADE (object, prescript)
2639     local fd_type = prescript and prescript.mplibfadetype
2640     local fd_stop = prescript and prescript.mplibfadestate
2641     if not fd_type then
2642         return fd_stop -- returns "stop" (if picture) or nil
2643     end
2644     local on, os, new
2645     if fd_type == "masking" then
2646         local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
2647         on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
2648         local bc = prescript.mplibmaskingbgcolor
2649         bc = bc and bc:gsub(":", " ")
2650         bc = bc and ("BC[%s]"):format(bc):gsub(decimals, rmzeros) or ""
2651         os = format("<</SMask<</S/Luminosity/G %s%>>>>",
2652             pdfmode and format(pdfetcs.resfmt, on) or on, bc)
2653     else
2654         local bbox = prescript.mplibfadebbox:explode":"
2655         local dx, dy = -bbox[1], -bbox[2]
2656         local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
2657         if not vec then
2658             if fd_type == "linear" then
2659                 vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
2660             else
2661                 local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
2662                 vec = {centerx, centery, centerx, centery} -- center for both circles
2663             end
2664         end
2665         local coords = { vec[1]+dx, vec[2]+dy, vec[3]+dx, vec[4]+dy }
2666         if fd_type == "linear" then
2667             coords = format("%f %f %f %f", tableunpack(coords))
2668         elseif fd_type == "circular" then
2669             local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
2670             local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":"
2671             tableinsert(coords, 3, radius[1])
2672             tableinsert(coords, radius[2])
2673             coords = format("%f %f %f %f %f %f", tableunpack(coords))
2674         else
2675             err("unknown fading method '%s'", fd_type)
2676         end
2677         fd_type = fd_type == "linear" and 2 or 3
2678         local opaq = (prescript.mplibfadeopacity or "1:0"):explode":"
2679         on = sh_pdfpageresources(fd_type, "0 1", "/DeviceGray", {{opaq[1]}}, {{opaq[2]}}, coords, 1)

```

```

2680 os = format("<</PatternType 2/Shading %s>>", format(pdfetcs.resfmt, on))
2681 on = update_pdfobjs(os)
2682 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
2683 local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
2684 :gsub(decimals,rmzeros)
2685 os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
2686 on = update_pdfobjs(os)
2687 local resources = format(pdfetcs.resfmt, on)
2688 on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
2689 local attr = tableconcat{
2690     "/Subtype/Form",
2691     "/BBox[" .. bbox .. "]",
2692     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
2693     "/Resources " .. resources,
2694     "/Group " .. format(pdfetcs.resfmt, on),
2695 } :gsub(decimals,rmzeros)
2696 on = update_pdfobjs(attr, streamtext)
2697 os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
2698 end
2699 on, new = update_pdfobjs(os)
2700 local key = add_extgs_resources(on,new)
2701 start_pdf_code()
2702 pdf_literalcode("/%s gs", key)
2703 if fd_stop then return "standalone" end
2704 return "start"
2705 end
2706

```

Transparency Group

```

2707 pdfetcs.tr_group = { shifts = { } }
2708 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
2709 local function do_preobj_GRP (object, prescript)
2710     local grstate = prescript and prescript.gr_state
2711     if not grstate then return end
2712     local trgroup = pdfetcs.tr_group
2713     if grstate == "start" then
2714         trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
2715         trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
2716         for _,v in ipairs(prescript.gr_type:explode",+") do
2717             trgroup[v] = true
2718         end
2719         trgroup.bbox = prescript.mplibgroupbbox:explode": "
2720         put2output[["\beginpgroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
2721     elseif grstate == "stop" then
2722         local llx,lly,urx,ury = tableunpack(trgroup.bbox)
2723         put2output(tableconcat{
2724             "\\egroup",
2725             format("\\wd\mplibscratchbox %fbp", urx-llx),
2726             format("\\ht\mplibscratchbox %fbp", ury-lly),

```

```

2727     "\\dp\\mplibscratchbox 0pt",
2728   })
2729   local grattr
2730   if trgroup.off then
2731     grattr = ""
2732   else
2733     local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>>",
2734                                     trgroup.isolated, trgroup.knockout))
2735     grattr = format("/Group %s", pdfetcs.resfmt:format(on))
2736   end
2737   local res = gather_resources("")
2738   local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
2739   if pdfmode then
2740     put2output(tableconcat{
2741       "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
2742       "/BBox[" .. bbox .. "]", grattr, "} resources{" .. res .. "}" .. "\\mplibscratchbox",
2743       "\\luamplibtagasgroupput{" .. trgroup.name .. "}" ..
2744       "[[\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}]]",
2745       "[\\wd\\mplibscratchbox 0pt\\ht\\mplibscratchbox 0pt\\dp\\mplibscratchbox 0pt]",
2746       "[\\box\\mplibscratchbox]",
2747       "}\\endgroup",
2748       "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{" ..
2749       "\\setbox\\mplibscratchbox\\hbox{\\hskip", -llx, "bp\\raise", -lly, "bp\\hbox{" ..
2750       "\\useboxresource \\the\\lastsavedboxresourceindex",
2751       "}}\\wd\\mplibscratchbox", urx-llx, "bp\\ht\\mplibscratchbox", ury-lly, "bp",
2752       "\\box\\mplibscratchbox}",
2753     })
2754   else
2755     trgroup.cnt = (trgroup.cnt or 0) + 1
2756     local objname = format("@mplibtrgr%s", trgroup.cnt)
2757     put2output(tableconcat{
2758       "\\special{pdf:boxobj " .. objname .. " bbox " .. bbox .. "}",
2759       "\\unhbox\\mplibscratchbox",
2760       "\\special{pdf:put @resources <<", res, ">>}",
2761       "\\special{pdf:exobj <<", grattr, ">>}",
2762       "\\luamplibtagasgroupput{" .. trgroup.name .. "}" ..
2763       "\\special{pdf:uxobj " .. objname .. "}",
2764       "}\\endgroup",
2765     })
2766     token.set_macro("luamplib.group." .. trgroup.name, tableconcat{
2767       "\\setbox\\mplibscratchbox\\hbox{\\hskip", -llx, "bp\\raise", -lly, "bp\\hbox{" ..
2768       "\\special{pdf:uxobj " .. objname .. "}",
2769       "}}\\wd\\mplibscratchbox", urx-llx, "bp\\ht\\mplibscratchbox", ury-lly, "bp",
2770       "\\box\\mplibscratchbox",
2771     }, "global")
2772   end
2773   trgroup.shifts[trgroup.name] = { llx, lly }
2774 end
2775 return grstate

```



```

2776 end
2777 function luamplib.registergroup (boxid, name, opts)
2778   local box = texgetbox(boxid)
2779   local wd, ht, dp = node.getwhd(box)
2780   local is_mask = opts.asgroup and opts.asgroup:find"masking"
2781   local res = opts.resources or ""
2782   res = gather_resources(res)
2783   local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
2784   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2785   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2786   if opts.matrix and opts.matrix:find"%a" then
2787     local data = format("mplibtransformmatrix(%s);", opts.matrix)
2788     process(data, "@mplibtransformmatrix")
2789     opts.matrix = format("%f %f %f %f %f %f", tableunpack(luamplib.transformmatrix))
2790   end
2791   local grtype = 3
2792   if opts.bbox then
2793     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2794     grtype = 2
2795   end
2796   local mpllx, mplly = get_macro'MPlllx', get_macro'MPlly'
2797   if is_mask then
2798     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
2799     t[5], t[6] = t[5]+mpllx, t[6]+mplly
2800     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
2801     mpllx, mplly = 0, 0
2802   end
2803   if opts.matrix then
2804     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
2805     grtype = opts.bbox and 4 or 1
2806   end
2807   if opts.asgroup and not opts.asgroup:find"off" then
2808     local t = { isolated = false, knockout = false, masking = false }
2809     for _, v in ipairs(opts.asgroup:explode",+") do t[v] = true end
2810     local on
2811     if t.masking then
2812       on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorsname or "/DeviceGray"))
2813     else
2814       local cs = opts.colorsname and ("/CS%s"):format(opts.colorsname) or ""
2815       on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
2816     end
2817     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
2818   end
2819   local trgroup = pdfetcs.tr_group
2820   trgroup.shifts[name] = { mpllx, mplly }
2821   local whd
2822   if pdfmode then
2823     attr = tableconcat(attr) :gsub(decimals, rmzeros)
2824     local index = tex.saveboxresource(boxid, attr, res, true, grtype)

```

```

2825 token.set_macro("luamplib.group"..name, tableconcat{
2826     "\\useboxresource ", index,
2827 }, "global")
2828 whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
2829 else
2830     trgroup.cnt = (trgroup.cnt or 0) + 1
2831     local objname = format("@mplibtrgr%s", trgroup.cnt)
2832     texsprint {
2833         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2834         "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2835         "\\hbox{\\unhbox ", boxid, "\\luamplibatnextshipout{",
2836         "\\special{pdf:bcontent}",
2837         "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
2838         "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2839         "\\special{pdf:put @resources <<", res, ">>}",
2840         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2841         "\\special{pdf:econtent}}",
2842     }
2843     token.set_macro("luamplib.group"..name, tableconcat{
2844         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
2845         "\\wd\\mplibscratchbox ", wd, "sp",
2846         "\\ht\\mplibscratchbox ", ht, "sp",
2847         "\\dp\\mplibscratchbox ", dp, "sp",
2848         "\\box\\mplibscratchbox",
2849     }, "global")
2850     whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
2851 end
2852 info("w/h/d of group '%s': %s", name, whd)
2853 end
2854

```

luamplib.convert: flushing figures

```

2855 do
2856     local function stop_special_effects(fade,opaq,over)
2857         if fade then -- fading
2858             stop_pdf_code()
2859         end
2860         if opaq then -- opacity
2861             pdf_literalcode(opaq)
2862         end
2863         if over then -- color
2864             if over:find"pdf:bc" then
2865                 put2output"\\special{pdf:ec}"
2866             else
2867                 put2output"\\special{color pop}"
2868             end
2869         end
2870     end
2871

```

For parsing prescript materials.

```

2872 local function script2table(s)
2873   local t = {}
2874   for _,i in ipairs(s:explode("\13+")) do
2875     local k,v = i:match("(.)=(.*)") -- v may contain = or empty.
2876     if k and v and k ~= "" and not t[k] then
2877       t[k] = v
2878     end
2879   end
2880   return t
2881 end
2882

```

Codes below to insert PDF lieterals are mostly from ConT_EXt general, with small changes when needed.

```

2883 local function pdf_textfigure(font,size,text,width,height,depth)
2884   text = text:gsub(".",function(c)
2885     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
2886   end)
2887   put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
2888 end
2889
2890 local bend_tolerance = 131/65536
2891
2892 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
2893
2894 local function pen_characteristics(object)
2895   local t = mplib.pen_info(object)
2896   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
2897   divider = sx*sy - rx*ry
2898   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
2899 end
2900
2901 local function concat(px, py) -- no tx, ty here
2902   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
2903 end
2904
2905 local function curved(ith,pth)
2906   local d = pth.left_x - ith.right_x
2907   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
2908     abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
2909     d = pth.left_y - ith.right_y
2910     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
2911       abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
2912       return false
2913     end
2914   end
2915   return true
2916 end

```

```

2917
2918 local function flushnormalpath(path,open)
2919     local pth, ith
2920     for i=1,#path do
2921         pth = path[i]
2922         if not ith then
2923             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
2924         elseif curved(ith,pth) then
2925             pdf_literalcode("%f %f %f %f %f %f c",
2926                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
2927         else
2928             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
2929         end
2930         ith = pth
2931     end
2932     if not open then
2933         local one = path[1]
2934         if curved(pth,one) then
2935             pdf_literalcode("%f %f %f %f %f %f c",
2936                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
2937         else
2938             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2939         end
2940     elseif #path == 1 then -- special case .. draw point
2941         local one = path[1]
2942         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2943     end
2944 end
2945
2946 local function flushconcatpath(path,open)
2947     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
2948     local pth, ith
2949     for i=1,#path do
2950         pth = path[i]
2951         if not ith then
2952             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
2953         elseif curved(ith,pth) then
2954             local a, b = concat(ith.right_x,ith.right_y)
2955             local c, d = concat(pth.left_x,pth.left_y)
2956             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
2957         else
2958             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
2959         end
2960         ith = pth
2961     end
2962     if not open then
2963         local one = path[1]
2964         if curved(pth,one) then
2965             local a, b = concat(pth.right_x,pth.right_y)

```

```

2966     local c, d = concat(one.left_x,one.left_y)
2967     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
2968   else
2969     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2970   end
2971 elseif #path == 1 then -- special case .. draw point
2972   local one = path[1]
2973   pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2974 end
2975 end
2976

```

Finally, flush figures by inserting PDF literals.

```

2977 local function flush (result,flusher)
2978   if result then
2979     local figures = result.fig
2980     if figures then
2981       for f=1, #figures do
2982         info("flushing figure %s",f)
2983         local figure = figures[f]
2984         local objects = figure:objects()
2985         local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
2986         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2987         local bbox = figure:boundingbox()
2988         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
2989         if urx < llx then

```

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

```

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

2990     else

```

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

```

2991     if tex_code_pre_mplib[f] then
2992       put2output(tex_code_pre_mplib[f])
2993     end
2994     pdf_startfigure(fignum,llx,lly,urx,ury)
2995     start_pdf_code()
2996     if objects then
2997       local savedpath = nil
2998       local savedhtap = nil
2999       for o=1,#objects do
3000         local object      = objects[o]
3001         local objecttype  = object.type

```

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

```

3002         local prescript  = object.prescript

```

```

3003     prescript = prescript and script2table(prescript) -- prescript is now a table
3004     local cr_over = do_preobj_CR(object,prescript) -- color
3005     local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3006     local fading_ = do_preobj_FADE(object,prescript) -- fading
3007     local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
3008     local shading_ = do_preobj_shading(object,prescript) -- shading pattern
3009     local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3010     if prescript and prescript.mplibtexboxid then
3011         put_tex_boxes(object,prescript)
3012     elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3013     elseif objecttype == "start_clip" then
3014         local evenodd = not object.istext and object.postscript == "evenodd"
3015         start_pdf_code()
3016         flushnormalpath(object.path,false)
3017         pdf_literalcode(evenodd and "W* n" or "W n")
3018     elseif objecttype == "stop_clip" then
3019         stop_pdf_code()
3020         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3021     elseif objecttype == "special" then

```

Collect T_EX codes that will be executed after flushing. Legacy behavior.

```

3022         if prescript and prescript.postmplibverbtx then
3023             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
3024         end
3025     elseif objecttype == "text" then
3026         local ot = object.transform -- 3,4,5,6,1,2
3027         start_pdf_code()
3028         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3029         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3030         stop_pdf_code()
3031     elseif not trgroup and fading_ ~= "stop" then
3032         local evenodd, collect, both = false, false, false
3033         local postscript = object.postscript
3034         if not object.istext then
3035             if postscript == "evenodd" then
3036                 evenodd = true
3037             elseif postscript == "collect" then
3038                 collect = true
3039             elseif postscript == "both" then
3040                 both = true
3041             elseif postscript == "eoboth" then
3042                 evenodd = true
3043                 both = true
3044             end
3045         end
3046         if collect then
3047             if not savedpath then
3048                 savedpath = { object.path or false }
3049                 savedhtap = { object.htap or false }

```

```

3050         else
3051             savedpath[#savedpath+1] = object.path or false
3052             savedhtap[#savedhtap+1] = object.htap or false
3053         end
3054     else
Removed from ConTeXt general: color stuff.
3055         local ml = object.miterlimit
3056         if ml and ml ~= miterlimit then
3057             miterlimit = ml
3058             pdf_literalcode("%f M",ml)
3059         end
3060         local lj = object.linejoin
3061         if lj and lj ~= linejoin then
3062             linejoin = lj
3063             pdf_literalcode("%i j",lj)
3064         end
3065         local lc = object.linecap
3066         if lc and lc ~= linecap then
3067             linecap = lc
3068             pdf_literalcode("%i J",lc)
3069         end
3070         local dl = object.dash
3071         if dl then
3072             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3073             if d ~= dashed then
3074                 dashed = d
3075                 pdf_literalcode(dashed)
3076             end
3077         elseif dashed then
3078             pdf_literalcode("[[] 0 d")
3079             dashed = false
3080         end
3081         local path = object.path
3082         local transformed, penwidth = false, 1
3083         local open = path and path[1].left_type and path[#path].right_type
3084         local pen = object.pen
3085         if pen then
3086             if pen.type == 'elliptical' then
3087                 transformed, penwidth = pen_characteristics(object) -- boolean, value
3088                 pdf_literalcode("%f w",penwidth)
3089                 if objecttype == 'fill' then
3090                     objecttype = 'both'
3091                 end
3092             else -- calculated by mplib itself
3093                 objecttype = 'fill'
3094             end
3095         end
end

```

Added : shading

```

3096     local shade_no, shade_stroking = do_preobj_SH(object,prescript) -- shading
3097     if shade_no then
3098         pdf_literalcode"q /Pattern cs"
3099         objecttype = false
3100     end
3101     if transformed then
3102         start_pdf_code()
3103     end
3104     if path then
3105         if savedpath then
3106             for i=1,#savedpath do
3107                 local path = savedpath[i]
3108                 if transformed then
3109                     flushconcatpath(path,open)
3110                 else
3111                     flushnormalpath(path,open)
3112                 end
3113             end
3114             savedpath = nil
3115         end
3116         if transformed then
3117             flushconcatpath(path,open)
3118         else
3119             flushnormalpath(path,open)
3120         end
3121         if objecttype == "fill" then
3122             pdf_literalcode(evenodd and "h f*" or "h f")
3123         elseif objecttype == "outline" then
3124             if both then
3125                 pdf_literalcode(evenodd and "h B*" or "h B")
3126             else
3127                 pdf_literalcode(open and "S" or "h S")
3128             end
3129         elseif objecttype == "both" then
3130             pdf_literalcode(evenodd and "h B*" or "h B")
3131         end
3132     end
3133     if transformed then
3134         stop_pdf_code()
3135     end
3136     local path = object.htap

```

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

```

3137     if path then
3138         if transformed then
3139             start_pdf_code()
3140         end
3141         if savedhtap then
3142             for i=1,#savedhtap do

```



```

3143         local path = savedhtap[i]
3144         if transformed then
3145             flushconcatpath(path,open)
3146         else
3147             flushnormalpath(path,open)
3148         end
3149     end
3150     savedhtap = nil
3151     evenodd = true
3152 end
3153 if transformed then
3154     flushconcatpath(path,open)
3155 else
3156     flushnormalpath(path,open)
3157 end
3158 if objecttype == "fill" then
3159     pdf_literalcode(evenodd and "h f*" or "h f")
3160 elseif objecttype == "outline" then
3161     pdf_literalcode(open and "S" or "h S")
3162 elseif objecttype == "both" then
3163     pdf_literalcode(evenodd and "h B*" or "h B")
3164 end
3165 if transformed then
3166     stop_pdf_code()
3167 end
3168 end

```

Added to ConT_EXt general: post-object colors and shading stuff. Beware q ... Q scope.

```

3169         if shade_no then -- shading
3170             pdf_literalcode("W%s %s /MPLibSh%s sh Q",
3171                 evenodd and "*" or "", shade_stroking and "s" or "n", shade_no)
3172         end
3173     end
3174 end
3175 if fading_ == "start" then
3176     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}
3177 elseif trgroup == "start" then
3178     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
3179 elseif fading_ == "stop" then
3180     local se = pdfetcs.fading.specialeffects
3181     if se then stop_special_effects(se[1], se[2], se[3]) end
3182 elseif trgroup == "stop" then
3183     local se = pdfetcs.tr_group.specialeffects
3184     if se then stop_special_effects(se[1], se[2], se[3]) end
3185 else
3186     stop_special_effects(fading_, tr_opaq, cr_over)
3187 end
3188 if fading_ or trgroup then -- extgs resetted
3189     miterlimit, linecap, linejoin, dashed = -1, -1, -1, false

```

```

3190         end
3191     end
3192 end
3193 stop_pdf_code()
3194 pdf_stopfigure()
output collected materials to PDF, plus legacy verbatimtex code.
3195     for _,v in ipairs(figcontents) do
3196         if type(v) == "table" then
3197             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3198         else
3199             texsprint(v)
3200         end
3201     end
3202     if #figcontents.post > 0 then texsprint(figcontents.post) end
3203     figcontents = { post = { } }
3204 end
3205 end
3206 end
3207 end
3208 end
3209
3210 function luamplib.convert (result, flusher)
3211     flush(result, flusher)
3212     return true -- done
3213 end
3214 end
3215
3216 function luamplib.colorconverter (cr)
3217     local n = #cr
3218     if n == 4 then
3219         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3220         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3221     elseif n == 3 then
3222         local r, g, b = cr[1], cr[2], cr[3]
3223         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3224     else
3225         local s = cr[1]
3226         return format("%.3f g %.3f G",s,s), "0 g 0 G"
3227     end
3228 end

```

2.2 T_EX package

First we need to load some packages.

```
3229 \ifcsname ProvidesPackage\endcsname
```

We need [T_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.

```

3230 \NeedsTeXFormat{LaTeX2e}
3231 \ProvidesPackage{luamplib}
3232 [2026/04/28 v2.41.0 mplib package for LuaTeX]
3233 \fi
3234 \ifdefined\newluafunction\else
3235 \input ltluatex
3236 \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 \LaTeX kernel. In Plain, atbegshi.sty is loaded.

```

3237 \ifnum\outputmode=0
3238 \ifdefined\AddToHookNext
3239 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3240 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3241 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3242 \else
3243 \input atbegshi.sty
3244 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3245 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3246 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3247 \fi
3248 \fi

```

Loading of lua code.

```

3249 \directlua{require("luamplib")}

```

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

```

3250 \ifx\pdfoutput\undefined
3251 \let\pdfoutput\outputmode
3252 \fi
3253 \ifx\pdfliteral\undefined
3254 \protected\def\pdfliteral{\pdfextension literal}
3255 \fi

```

Set the format for METAPOST.

```

3256 \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.

```

3257 \ifnum\pdfoutput>0
3258 \let\mplibtoPDF\pdfliteral
3259 \else
3260 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3261 \ifcsname PackageInfo\endcsname
3262 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3263 \else
3264 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3265 \fi
3266 \fi

```

To make `mplibcode` typeset always in horizontal mode.

```
3267 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3268 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3269 \mplibnoforcehmode
```

Catcode. We want to allow comment sign in `mplibcode`.

```
3270 \def\mplibsetupcatcodes{%
3271   %catcode`\{=12 %catcode`\}=12
3272   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3273   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
3274 }
```

Make `btex...etex` box zero-metric.

```
3275 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}
```

use Transparency Group

```
3276 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3277 \def\usemplibgroupmain#1{%
3278   \prependtomplibbox\hbox dir TLT\bgroup
3279   \csname luamplib.group.#1\endcsname
3280   \egroup
3281 }
3282 \protected\def\mplibgroup#1{%
3283   \begingroup
3284   \def\MPllx{0}\def\MPlly{0}%
3285   \def\mplibgroupname{#1}%
3286   \mplibgroupgetnexttok
3287 }
3288 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3289 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
3290 \def\mplibgroupbranch{%
3291   \ifx [\nexttok
3292     \expandafter\mplibgroupopts
3293   \else
3294     \ifx\mplibsptoken\nexttok
3295       \expandafter\expandafter\expandafter\mplibgroupskipspace
3296     \else
3297       \let\mplibgroupoptions\empty
3298       \expandafter\expandafter\expandafter\mplibgroupmain
3299     \fi
3300   \fi
3301 }
3302 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3303 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3304 \protected\def\endmplibgroup{\egroup
3305   \directlua{ luamplib.registergroup(
3306     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3307   )}%
3308   \endgroup
3309 }
```

Patterns

```

3310 {\def\:{\global\let\mplibsptoken= } \: }
3311 \protected\def\mppattern#1{%
3312   \begingroup
3313   \def\MPllx{0}\def\MPlly{0}%
3314   \def\mplibpatternname{#1}%
3315   \mplibpatterngetnexttok
3316 }
3317 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3318 \def\mplibpatternskipspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3319 \def\mplibpatternbranch{%
3320   \ifx [\nexttok
3321     \expandafter\mplibpatternopts
3322   \else
3323     \ifx\mplibsptoken\nexttok
3324       \expandafter\expandafter\expandafter\mplibpatternskipspace
3325     \else
3326       \let\mplibpatternoptions\empty
3327       \expandafter\expandafter\expandafter\mplibpatternmain
3328     \fi
3329   \fi
3330 }
3331 \def\mplibpatternopts[#1]{%
3332   \def\mplibpatternoptions{#1}%
3333   \mplibpatternmain
3334 }
3335 \def\mplibpatternmain{%
3336   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3337 }
3338 \protected\def\endmppattern{%
3339   \egroup
3340   \directlua{ luamplib.registerpattern(
3341     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3342   )}%
3343   \endgroup
3344 }

```

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

```

3345 \def\mpfiginstancename{@mpfig}
3346 \protected\def\mpfig{%
3347   \begingroup
3348   \futurelet\nexttok\mplibmpfigbranch
3349 }
3350 \def\mplibmpfigbranch{%
3351   \ifx *\nexttok
3352     \expandafter\mplibprempfig
3353   \else
3354     \ifx [\nexttok
3355       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig

```

```

3356 \else
3357 \expandafter\expandafter\expandafter\mplibmainmpfig
3358 \fi
3359 \fi
3360 }
3361 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
3362 \def\mplibmainmpfig{%
3363 \begingroup
3364 \mplibsetupcatcodes
3365 \mplibdomainmpfig
3366 }
3367 \long\def\mplibdomainmpfig#1\endmpfig{%
3368 \endgroup
3369 \directlua{
3370 local legacy = luamplib.legacyverbatim
3371 local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3372 local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3373 luamplib.legacyverbatim = false
3374 luamplib.everymplib["\mpfiginstancename"] = ""
3375 luamplib.everyendmplib["\mpfiginstancename"] = ""
3376 luamplib.process_mplibcode(
3377 "beginfig(0) "..everympfig.." "..[unexpanded{#1}]===".." ..everyendmpfig.." endfig;",
3378 "\mpfiginstancename")
3379 luamplib.legacyverbatim = legacy
3380 luamplib.everymplib["\mpfiginstancename"] = everympfig
3381 luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3382 }%
3383 \endgroup
3384 }
3385 \def\mplibprempfig#1{%
3386 \begingroup
3387 \mplibsetupcatcodes
3388 \mplibdopremfig
3389 }
3390 \long\def\mplibdopremfig#1\endmpfig{%
3391 \endgroup
3392 \directlua{
3393 local legacy = luamplib.legacyverbatim
3394 local everympfig = luamplib.everymplib["\mpfiginstancename"]
3395 local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3396 luamplib.legacyverbatim = false
3397 luamplib.everymplib["\mpfiginstancename"] = ""
3398 luamplib.everyendmplib["\mpfiginstancename"] = ""
3399 luamplib.process_mplibcode([unexpanded{#1}]==="","\mpfiginstancename")
3400 luamplib.legacyverbatim = legacy
3401 luamplib.everymplib["\mpfiginstancename"] = everympfig
3402 luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3403 }%
3404 \endgroup

```

```

3405 }
3406 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3407 \unless\ifcsname ver@luamplib.sty\endcsname
3408 \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3409 \protected\def\mplibcode{%
3410 \begingroup
3411 \futurelet\nexttok\mplibcodebranch
3412 }
3413 \def\mplibcodebranch{%
3414 \ifx \nexttok
3415 \expandafter\mplibcodegetinstancename
3416 \else
3417 \global\let\currentmpinstancename\empty
3418 \expandafter\mplibcodeindeed
3419 \fi
3420 }
3421 \def\mplibcodeindeed{%
3422 \begingroup
3423 \mplibsetupcatcodes
3424 \mplibdocode
3425 }
3426 \long\def\mplibdocode#1\endmplibcode{%
3427 \endgroup
3428 \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]==],"\currentmpinstancename")}%
3429 \endgroup
3430 }
3431 \protected\def\endmplibcode{endmplibcode}
3432 \else

```

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

```

3433 \newenvironment{mplibcode}[1][{}%
3434 \xdef\currentmpinstancename{#1}%
3435 \mplibtmptoks{}\ltxdomplibcode
3436 }{}
3437 \def\ltxdomplibcode{%
3438 \begingroup
3439 \mplibsetupcatcodes
3440 \ltxdomplibcodeindeed
3441 }
3442 \def\mplib@mplibcode{mplibcode}
3443 \long\def\ltxdomplibcodeindeed#1\end#2{%
3444 \endgroup
3445 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3446 \def\mplibtemp@a{#2}%
3447 \ifx\mplib@mplibcode\mplibtemp@a
3448 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]==],"\currentmpinstancename")}%
3449 \end{mplibcode}%
3450 \else

```

```

3451 \mplibmptoks\expandafter{\the\mplibmptoks\end{#2}}%
3452 \expandafter\ltxdomplibcode
3453 \fi
3454 }
3455 \fi

User settings.
3456 \def\mplibshowlog#1{\directlua{
3457   local s = string.lower("#1")
3458   if s == "enable" or s == "true" or s == "yes" then
3459     luampLib.showlog = true
3460   else
3461     luampLib.showlog = false
3462   end
3463 }}
3464 \def\mpliblegacybehavior#1{\directlua{
3465   local s = string.lower("#1")
3466   if s == "enable" or s == "true" or s == "yes" then
3467     luampLib.legacyverbatim = true
3468   else
3469     luampLib.legacyverbatim = false
3470   end
3471 }}
3472 \def\mplibverbatim#1{\directlua{
3473   local s = string.lower("#1")
3474   if s == "enable" or s == "true" or s == "yes" then
3475     luampLib.verbatiminput = true
3476   else
3477     luampLib.verbatiminput = false
3478   end
3479 }}
3480 \newtoks\mplibmptoks

\everymplib & \everyendmplib: macros resetting luampLib.every(end)mpLib tables
3481 \ifcsname ver@luampLib.sty\endcsname
3482 \protected\def\everymplib{%
3483   \begingroup
3484   \mplibsetupcatcodes
3485   \mplibdoeverymplib
3486 }
3487 \protected\def\everyendmplib{%
3488   \begingroup
3489   \mplibsetupcatcodes
3490   \mplibdoeveryendmplib
3491 }
3492 \newcommand\mplibdoeverymplib[2][]{%
3493   \endgroup
3494   \directlua{
3495     luampLib.everymplib["#1"] = [===[\unexpanded{#2}]===[
3496   ]%

```



```

3497 }
3498 \newcommand\mplibdoeveryendmplib[2][]{%
3499   \endgroup
3500   \directlua{
3501     luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===[
3502   ]%
3503 }
3504 \else
3505   \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3506   \protected\def\everymplib#1#{%
3507     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3508     \begingroup
3509     \mplibsetupcatcodes
3510     \mplibdoeverymplib
3511   }
3512   \long\def\mplibdoeverymplib#1{%
3513     \endgroup
3514     \directlua{
3515       luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===[
3516     ]%
3517   }
3518   \protected\def\everyendmplib#1#{%
3519     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3520     \begingroup
3521     \mplibsetupcatcodes
3522     \mplibdoeveryendmplib
3523   }
3524   \long\def\mplibdoeveryendmplib#1{%
3525     \endgroup
3526     \directlua{
3527       luamplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===[
3528     ]%
3529   }
3530 \fi

```

TeX macros for dimen/color

```

3531 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
3532 \def\mpcolor#1#{\domplibcolor{#1}}
3533 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

3534 \def\mplibnumbersystem#1{\directlua{
3535   local t = "#1"
3536   if t == "binary" then t = "decimal" end
3537   luamplib.numbersystem = t
3538 }}

```

Settings for .mp cache files.

```

3539 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
3540 \def\mplibdomakenocache#1,{%

```

```

3541 \ifx\empty#1\empty
3542 \expandafter\mplibdomakenocache
3543 \else
3544 \ifx\stop#1\else
3545 \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
3546 \expandafter\expandafter\expandafter\mplibdomakenocache
3547 \fi
3548 \fi
3549 }
3550 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
3551 \def\mplibdocancelnocache#1,{%
3552 \ifx\empty#1\empty
3553 \expandafter\mplibdocancelnocache
3554 \else
3555 \ifx\stop#1\else
3556 \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
3557 \expandafter\expandafter\expandafter\mplibdocancelnocache
3558 \fi
3559 \fi
3560 }
3561 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

3562 \def\mplibtexttextlabel#1{\directlua{
3563   local s = string.lower("#1")
3564   if s == "enable" or s == "true" or s == "yes" then
3565     luamplib.texttextlabel = true
3566   else
3567     luamplib.texttextlabel = false
3568   end
3569 }}
3570 \def\mplibcodeinherit#1{\directlua{
3571   local s = string.lower("#1")
3572   if s == "enable" or s == "true" or s == "yes" then
3573     luamplib.codeinherit = true
3574   else
3575     luamplib.codeinherit = false
3576   end
3577 }}
3578 \def\mplibglobaltexttext#1{\directlua{
3579   local s = string.lower("#1")
3580   if s == "enable" or s == "true" or s == "yes" then
3581     luamplib.globaltexttext = true
3582   else
3583     luamplib.globaltexttext = false
3584   end
3585 }}

```

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

```
3586 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi
```

We encapsulate the literals.

```
3587 \def\mplibstarttoPDF#1#2#3#4{%
3588   \prependtomplibbox
3589   \hbox dir TLT\bgroup
3590   \xdef\MPllx{#1}\xdef\MPlly{#2}%
3591   \xdef\MPurx{#3}\xdef\MPury{#4}%
3592   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3593   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3594   \parskip0pt%
3595   \leftskip0pt%
3596   \parindent0pt%
3597   \everypar{}%
3598   \setbox\mplibscratchbox\ vbox\bgroup
3599   \noindent
3600 }
3601 \def\mplibstoptoPDF{%
3602   \par
3603   \egroup %
3604   \setbox\mplibscratchbox\ hbox %
3605     {\hskip-\MPllx bp%
3606      \raise-\MPlly bp%
3607      \box\mplibscratchbox}%
3608   \setbox\mplibscratchbox\ vbox to \MPheight
3609     {\vfill
3610      \hsize\MPwidth
3611      \wd\mplibscratchbox0pt%
3612      \ht\mplibscratchbox0pt%
3613      \dp\mplibscratchbox0pt%
3614      \box\mplibscratchbox}%
3615   \wd\mplibscratchbox\MPwidth
3616   \ht\mplibscratchbox\MPheight
3617   \box\mplibscratchbox
3618   \egroup
3619 }
```

Text items have a special handler.

```
3620 \def\mplibtexttext#1#2#3#4#5{%
3621   \begingroup
3622   \setbox\mplibscratchbox\ hbox
3623     {\font\temp=#1 at #2bp%
3624      \temp
3625      #3}%
3626   \setbox\mplibscratchbox\ hbox
3627     {\hskip#4 bp%
3628      \raise#5 bp%
3629      \box\mplibscratchbox}%
3630   \wd\mplibscratchbox0pt%
3631   \ht\mplibscratchbox0pt%
```

```

3632 \dp\mplibscratchbox0pt%
3633 \box\mplibscratchbox
3634 \endgroup
3635 }

```

Input luamplib.cfg when it exists.

```

3636 \openin0=luamplib.cfg
3637 \ifeof0 \else
3638 \closein0
3639 \input luamplib.cfg
3640 \fi

```

Code for tagpdf

```

3641 \def\luamplibtagtextboxset#1#2{#2}
3642 \let\luamplibnotagtextboxset\luamplibtagtextboxset
3643 \let\luamplibtagasgroupset\relax
3644 \let\luamplibtagasgroupput\luamplibtagtextboxset
3645 \ifcsname SuspendTagging\endcsname\else\endinput\fi
3646 \ifcsname ver@tagpdf.sty\endcsname \else
3647 \ExplSyntaxOn
3648 \keys_define:nn{luamplib/tagging}
3649 {
3650 ,alt .code:n = { }
3651 ,actualtext .code:n = { }
3652 ,artifact .code:n = { }
3653 ,text .code:n = { }
3654 ,off .code:n = { }
3655 ,tag .code:n = { }
3656 ,adjust-BBox .code:n = { }
3657 ,tagging-setup .code:n = { }
3658 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3659 ,instancename .meta:n = { instance = {#1} }
3660 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
3661 }
3662 \RenewDocumentCommand\mplibcode{0{}}
3663 {
3664 \tl_gclear:N \currentmpinstancename
3665 \keys_set:ne{luamplib/tagging}{#1}
3666 \mplibmptoks{}\ltxdomplibcode
3667 }
3668 \cs_set_eq:NN \mplibaltext \use_none:n
3669 \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 L^AT_EX. 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 L^AT_EX's \everypar should have done.

```

3670 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
3671 \ExplSyntaxOff

```

```

3672 \endinput\fi
3673 \ExplSyntaxOn
3674 \tl_new:N \l__luamplib_tag_envname_tl
3675 \tl_new:N \l__luamplib_tag_alt_tl
3676 \tl_new:N \l__luamplib_tag_alt_dflt_tl
3677 \tl_new:N \l__luamplib_tag_actual_tl
3678 \tl_new:N \l__luamplib_tag_struct_tl
3679 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
3680 \bool_new:N \l__luamplib_tag_usetext_bool
3681 \bool_new:N \l__luamplib_tag_bboxcorr_bool
3682 \seq_new:N \l__luamplib_tag_bboxcorr_seq
3683 \tl_new:N \l__luamplib_tag_bbox_draw_tl
3684 \tl_new:N \l__luamplib_BBoxllx_tl
3685 \tl_new:N \l__luamplib_BBoxlly_tl
3686 \tl_new:N \l__luamplib_BBoxurx_tl
3687 \tl_new:N \l__luamplib_BBoxury_tl
3688 \msg_new:nnn {luamplib}{figure-text-reuse}
3689 {
3690   tex-text~box~#1~probably~is~incorrectly~tagged.~
3691   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
3692   Check~the~resulting~PDF.
3693 }
3694 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
3695 {
3696   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
3697   Using~mplibgroup~with~text~mode~is~not~recommended.~
3698   Check~the~resulting~PDF.
3699 }
3700 \msg_new:nnn {luamplib}{alt-text-missing}
3701 {
3702   Alternate~text~for~#1~is~missing.~
3703   Using~the~default~value~'#2'~instead.
3704 }

```

Sockets for tex-text boxes.

```

3705 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
3706 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
3707 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
3708 {

```

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.

```

3709 \bool_if:NTF \l__luamplib_tag_usetext_bool
3710 {
3711   \tag_mc_end_push:
3712   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
3713   \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.

```

3714   \tag_mc_begin:n{tag=text}

```

```

3715     #2
3716     \tag_mc_end:
3717     \tag_struct_end:
3718     \tag_mc_begin_pop:n{ }
3719   }
3720   {
3721     \tag_suspend:n{\luamplibtagtextboxset}
3722     #2
3723     \tag_resume:n{\luamplibtagtextboxset}
3724   }
3725 }
3726 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
3727 {
3728   \bool_lazy_and:nnTF
3729   { \l__luamplib_tag_usertext_bool }
3730   { \cs_if_free_p:c {luamplib.taggedbox.#1} }
3731   {
3732     \tag_resume:n{\mplibputtextbox}
3733     \tag_mc_end:
3734     \cs_if_exist:cTF {luamplib.taggedbox.#1}
3735     {
3736       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
3737       #2
3738       \cs_undefine:c {luamplib.taggedbox.#1}
3739     }
3740     {
3741       \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
3742       \tag_mc_begin:n{ }
3743       \int_set:Nn \l_tmpa_int {#1}
3744       \tag_mc_reset_box:N \l_tmpa_int
3745       #2
3746       \tag_mc_end:
3747     }
3748     \tag_mc_begin:n{artifact}
3749   }
3750   {
3751     \int_set:Nn \l_tmpa_int {#1}
3752     \tag_mc_reset_box:N \l_tmpa_int
3753     #2
3754   }
3755 }
3756 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
3757 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
3758 \cs_set_nopar:Npn \luamplibtagtextboxset
3759 {
3760   \tag_socket_use:nnn{luamplib/texttext/set}
3761 }

```

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

the artifact MC-chunks.

```
3762 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
3763 {
3764   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
3765   \bool_set_false:N \l__luamplib_tag_usetext_bool
3766   \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
3767   \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
3768   \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
3769 }
3770 \cs_set_nopar:Npn \mplibputtextbox #1
3771 {
3772   \vbox to 0pt{\vss\hbox to 0pt{
3773     \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}
3774     \hss}}
3775 }
```

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

```
3776 \cs_set_nopar:Npn \luamplibtagasgroupset
3777 {
3778   \bool_set_false:N \l__luamplib_tag_usetext_bool
3779 }
3780 \cs_set_nopar:Npn \luamplibtagasgroupput
3781 {
3782   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
3783   \tag_socket_use:nnn{luamplib/mplibgroup/put}
3784 }
```

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

```
3785 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
3786 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
3787 {
3788   \cs_if_free:cT {luamplib.mplibgroup.text.#1}
3789   {
3790     \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
3791     \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
3792   }
3793   \tag_mc_end:
3794   \tag_mc_begin:n{tag=text}
3795   #2
3796   \tag_mc_end:
3797   \tag_mc_begin:n{artifact}
3798 }
3799 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}
```

A macro for BBox attribute

```
3800 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
3801 {
3802   \tl_set:Ne \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
3803   \tex_savepos:D
```

```

3804 \property_record:ee{\l_tmpa_tl}{xpos,ypos}
3805 \tl_set:Ne \l__luamplib_BBox_llx_tl
3806 { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
3807 \tl_set:Ne \l__luamplib_BBox_lly_tl
3808 { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
3809 \tl_set:Ne \l__luamplib_BBox_urx_tl
3810 { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
3811 \tl_set:Ne \l__luamplib_BBox_ury_tl
3812 { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
3813 \bool_if:NT \l__luamplib_tag_bboxcorr_bool
3814 {
3815   \int_zero:N \l_tmpa_int
3816   \tl_map_inline:nn
3817   {
3818     \l__luamplib_BBox_llx_tl
3819     \l__luamplib_BBox_lly_tl
3820     \l__luamplib_BBox_urx_tl
3821     \l__luamplib_BBox_ury_tl
3822   }
3823   {
3824     \int_incr:N \l_tmpa_int
3825     \tl_set:Ne ##1
3826     {
3827       \fp_eval:n
3828       {
3829         ##1
3830         +
3831         \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
3832       }
3833     }
3834   }
3835 }
3836 \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
3837 {
3838   /O /Layout /BBox [
3839     \l__luamplib_BBox_llx_tl\c_space_tl
3840     \l__luamplib_BBox_lly_tl\c_space_tl
3841     \l__luamplib_BBox_urx_tl\c_space_tl
3842     \l__luamplib_BBox_ury_tl
3843   ]
3844 }
3845 \bool_if:NT \l__tag_graphic_debug_bool
3846 {
3847   \iow_log:e
3848   {
3849     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
3850     \l__luamplib_BBox_llx_tl\c_space_tl
3851     \l__luamplib_BBox_lly_tl\c_space_tl
3852     \l__luamplib_BBox_urx_tl\c_space_tl

```



```

3853     \l__luamplib_BBox_ury_tl
3854 }
3855 \sys_if_output_pdf:TF
3856 {
3857     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
3858     {
3859         \pdfextension save\relax
3860         \opacity_select:n{0.5} \color_select:n{red}
3861         \pdfextension literal~text
3862         {
3863             \l__luamplib_BBox_llx_tl\c_space_tl
3864             \l__luamplib_BBox_lly_tl\c_space_tl
3865             \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3866             \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3867             re~f
3868         }
3869         \pdfextension restore\relax
3870     }
3871 }
3872 {
3873     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
3874     {
3875         \special{pdf:bcontent}
3876         \opacity_select:n{0.5} \color_select:n{red}
3877         \special{pdf:code~
3878             1~0~0~1~
3879             -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
3880             -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
3881             cm
3882         }
3883         \special{pdf:code~
3884             \l__luamplib_BBox_llx_tl\c_space_tl
3885             \l__luamplib_BBox_lly_tl\c_space_tl
3886             \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3887             \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3888             re~f
3889         }
3890         \special{pdf:econtent}
3891     }
3892 }
3893 }
3894 }

```

Sockets for main process

```

3895 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
3896 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
3897 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
3898 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
3899 {

```

```

3900 \tag_mc_end_push:
3901 \tl_if_empty:NT\l__luamplib_tag_alt_tl
3902 {
3903   \tl_if_empty:eTF{#1}
3904   { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
3905   { \tl_set:Ne \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
3906   \msg_warning:nnVV{luamplib}{alt-text-missing}
3907   \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
3908 }
3909 \tag_struct_begin:n
3910 {
3911   tag=\l__luamplib_tag_struct_tl,
3912   alt=\l__luamplib_tag_alt_tl,
3913 }
3914 \tag_mc_begin:n{ }
3915 }
3916 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
3917 {
3918   \__luamplib_tag_bbox_attribute:n {#1}
3919   #2
3920   \tl_use:N \l__luamplib_tag_bbox_draw_tl
3921   \tag_mc_end:
3922   \tag_struct_end:
3923   \tag_mc_begin_pop:n{ }
3924 }
3925 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
3926 {
3927   \tag_mc_end_push:
3928   \tag_struct_begin:n
3929   {
3930     tag=Span,
3931     actualtext=\l__luamplib_tag_actual_tl,
3932   }
3933   \tag_mc_begin:n{ }
3934 }
3935 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
3936 {
3937   #2
3938   \tag_mc_end:
3939   \tag_struct_end:
3940   \tag_mc_begin_pop:n{ }
3941 }
3942 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
3943 {
3944   \tag_mc_end_push:
3945   \tag_mc_begin:n{artifact}
3946 }
3947 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
3948 {

```

```

3949 #2
3950 \tag_mc_end:
3951 \tag_mc_begin_pop:n{}
3952 }

```

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

```

3953 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
3954 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
3955 {
3956 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
3957 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
3958 }
3959 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
3960 {
3961 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
3962 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

3963 \prependtomplibbox \mplibnoforcehmode
3964 \mode_if_vertical:T { \noindent \aftergroup\par }
3965 }
3966 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
3967 {
3968 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
3969 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
3970 }
3971 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
3972 {
3973 \bool_set_true:N \l__luamplib_tag_usetext_bool
3974 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
3975 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
3976 \prependtomplibbox \mplibnoforcehmode
3977 \mode_if_vertical:T { \noindent \aftergroup\par }
3978 }
3979 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
3980 {
3981 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
3982 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
3983 }
3984 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```

3985 \keys_define:nn{luamplib/tagging}
3986 {
3987 ,alt .code:n =
3988 {
3989 \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
3990 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
3991 }

```

```

3992 ,actualtext .code:n =
3993 {
3994   \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
3995   \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
3996 }
3997 ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
3998 ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
3999 ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4000 ,tag .code:n =
4001 {
4002   \str_case:nnF {#1}
4003   {
4004     {false} { \keys_set:nn {luamplib/tagging} {off} }
4005     {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4006   }
4007   {
4008     \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4009     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4010   }
4011 }
4012 ,adjust-BBox .code:n =
4013 {
4014   \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4015   \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4016 }
4017 ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4018 }
4019 \keys_define:nn {luamplib/instance}
4020 {
4021   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4022   ,instancename .meta:n = { instance = {#1} }
4023   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4024 }

```

Redefine our macros

```

4025 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4026 {
4027   \prependtomplibbox
4028   \hbox dir~TLT\bgroup
4029     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4030     \xdef\MPlIx{#1}\xdef\MPlly{#2}%
4031     \xdef\MPlrx{#3}\xdef\MPlry{#4}%
4032     \xdef\MPlwidth{\the\dimexpr#3bp-#1bp\relax}%
4033     \xdef\MPlheight{\the\dimexpr#4bp-#2bp\relax}%
4034     \parskip0pt
4035     \leftskip0pt
4036     \parindent0pt
4037     \everypar{}%
4038     \setbox\mplibscratchbox\vbox\bgroup

```

```

4039     \tag_suspend:n{\mplibstarttoPDF}
4040     \noindent
4041 }
4042 \cs_set_nopar:Npn \mplibstoptoPDF
4043 {
4044     \par
4045     \egroup
4046     \setbox\mplibscratchbox\hbox
4047     {\hskip-\MPllx bp
4048      \raise-\MPlly bp
4049      \box\mplibscratchbox}%
4050     \setbox\mplibscratchbox\vbox to \MPheight
4051     {\vfill
4052      \hsize\MPwidth
4053      \wd\mplibscratchbox0pt
4054      \ht\mplibscratchbox0pt
4055      \dp\mplibscratchbox0pt
4056      \box\mplibscratchbox}%
4057     \wd\mplibscratchbox\MPwidth
4058     \ht\mplibscratchbox\MPheight
4059     \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4060     \egroup
4061 }
4062 \RenewDocumentCommand\mplibcode{0{}}
4063 {
4064     \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4065     \tl_gclear:N \currentmpinstancename
4066     \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4067     \keys_set:nV {luamplib/instance} \l_tmpa_tl
4068     \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4069     \tag_socket_use:n{luamplib/figure/init}
4070     \mplibtmptoks{}\ltxdomplibcode
4071 }
4072 \RenewDocumentCommand\mpfig{s 0{}}
4073 {
4074     \begingroup
4075     \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4076     \keys_set_known:ne {luamplib/tagging} {#2}
4077     \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4078     \tag_socket_use:n{luamplib/figure/init}
4079     \IfBooleanTF{#1} { \mplibprempfig * }
4080                     { \mplibmainmpfig }
4081 }
4082 \RenewDocumentCommand\usemplibgroup{0{ } m}
4083 {
4084     \begingroup
4085     \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4086     \keys_set_known:ne {luamplib/tagging} {#1}
4087     \tag_socket_use:n{luamplib/figure/init}

```

```

4088 \prependtomplibbox\hbox dir~TLT\bgroup
4089 \tag_socket_use:nn{luamplib/figure/begin}{#2}
4090 \setbox\mplibscratchbox\hbox\bgroup
4091 \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
4092 \tag_socket_use:nnn{luamplib/mpplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4093 \egroup
4094 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4095 \egroup
4096 \endgroup
4097 }

```

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

```

4098 \cs_new_nopar:Npn \mplibaltttext #1
4099 {
4100 \tl_set:Nc \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4101 }
4102 \cs_new_nopar:Npn \mplibactualtext #1
4103 {
4104 \tl_set:Nc \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4105 }
4106 \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 subroutine 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.