

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/09/01 V2.42.9

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	Forcing horizontal mode	3
1.1.2	Insert some code into every code chunk	3
1.1.3	Choosing a METAPOST format	3
1.1.4	Choosing a number system	4
1.1.5	Showing METAPOST log	4
1.1.6	verbatimtex Legacy behavior	5
1.1.7	$\TeX$ -text labels	5
1.1.8	Inherit METAPOST code	6
1.1.9	<code>\mplibglobaltexttext</code>	6
1.1.10	Separate METAPOST instances	6
1.1.11	Verbatim input	7
1.1.12	$\TeX$ dimen	7
1.1.13	$\TeX$ color	7
1.1.14	<code>\mpfig</code> , <code>\endmpfig</code>	8
1.1.15	About cache files	9
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	$\TeX$ dimen and color	11
1.2.2	More on $\TeX$ color	11
1.2.3	Fill and stroke colors	12
1.2.4	Transparency	12

1.2.5	Fill and stroke transparency	13
1.2.6	Text outline as a text image	13
1.2.7	Glyph outline	14
1.2.8	Drawing a glyph outline	14
1.2.9	Text outline as a path image	15
1.2.10	Unicode-aware length	15
1.2.11	Unicode-aware substring	15
1.2.12	Shading	16
1.2.13	Fading	18
1.2.14	Tiling pattern	19
1.2.15	Form XObject from METAPOST side	22
1.2.16	Form XObject from T _E X side	25
1.2.17	Masking	26
1.2.18	Free-form Gouraud-shaded triangle mesh shading	27
1.2.19	Lattice-form Gouraud-shaded triangle mesh shading	28
1.2.20	Coons patch mesh shading	29
1.2.21	Tensor-product patch mesh shading	31
1.3	Lua	32
1.3.1	runscript	32
1.3.2	Accessing METAPOST variables	33
1.3.3	Running a METAPOST code	33
1.3.4	Registering a tiling pattern	33
1.3.5	Registering a form XObject	34
2	Implementation	34
2.1	Lua module	34
2.2	T _E Xpackage	117
3	The GNU GPL License v2	138

1 Documentation

This package aims at providing a simple way to typeset directly METAPOST code in a document with LuaT_EX. LuaT_EX is built with the Lua mplib library, that runs METAPOST code. This package is basically a wrapper for the Lua mplib functions and some T_EX 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 L^AT_EX in the `mplibcode` environment.

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

The code of luamplib is basically from the `luatex-mp lib.lua` and `luatex-mp lib.tex` files from ConT_EXt. They have been adapted to L^AT_EX 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 <string>` 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 `DVIPDFMx` is the only DVI tool currently supported.

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

1.1 $\TeX$

1.1.1 Forcing horizontal mode

When `\mplibforcehmode` 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 Insert some code into every code chunk

`\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 Choosing a METAPOST format

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

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.

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 § 1.2.9). You can try other effects as well, though we did not fully test their proper functioning.

transparency (texdoc metafun § 8.2) Transparency is so simple that you can apply it to an object, with *plain* format as well as *metafun*, just by appending `withprescript "tr_transparency=<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 § 1.2.12.

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

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

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

Transparency group is available with *plain* format as well with extended functionality. See below § 1.2.15.

1.1.4 Choosing a number system

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

1.1.5 Showing METAPOST log

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`. Default value is `disable`.

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

1.1.6 verbatimtex Legacy behavior

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

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

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.
```



diameter: 22.62764bp.

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 $\TeX$ -text labels

`\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)`. Default value is `disable`.

³But the recommended way to reuse a figure is using `\mplibgroup` command. See below § 1.2.16.

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.

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

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

1.1.8 Inherit METAPOST code

`\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. Default value is `disable`.

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 $\TeX$ environment `mplibcode` or Plain $\TeX$ commands `\mplibcode ... \endmplibcode`. The syntax for $\TeX$ is:

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

The behavior is as follows.

- All the variables and functions are shared only among all the environments belonging to the same instance.
- `\mplibcodeinherit` only affects 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 Verbatim input

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. Default value is `disable`.

1.1.12 $\TeX$ `dimen`

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 $\TeX$ color

With the command `\mpcolor[...]{...}`, color 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 `color` or `xcolor`'s `\color` command. For spot colors, `l3color` module is well supported in PDF and DVI mode. Package `colorspace` is also supported in PDF mode, but could conflict with `luamplib`'s special features such as uncolored tiling pattern when `\DocumentMetadata`, i.e. 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. 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 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(withpen pencircle scaled 1 withcolor 2/3red); }
\mpfig* input boxes \endmpfig
\mpfig
  circleit.a(btex Box 1 etex); drawboxed(a);
\endmpfig
```



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 `⟨filename⟩` 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 $\TeX$'s `picture` environment (`texdoc latex-lab-graphic`). The default tagging mode is the `alt` key with Figure structure.

`alt=⟨text⟩` starts a Figure tag by default and sets an alternate text of the figure from the `⟨text⟩`. 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 "\mplibaltttext{<text>}";`

actualtext=`<text>` starts a Span tag implicitly and sets a replacement text (a.k.a. actual text) from the `<text>`. 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>}";`

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 mplibgraphicstext "$\sqrt{x}$" shifted 48down ;
  endfig;
\end{mplibcode}
```

$\sqrt{2}$
 $\sqrt{3}$
 $\sqrt{5}$
 $\sqrt{7}$
 $\sqrt{x}$

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.

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.

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 T_EX's regular math formula tagging, for which the `text` key is needed.

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

You can set these options anywhere in the document by declaring `\SetKeys[luamplib/tagging]{ $\langle$ key-val list $\rangle$ }`, which will affect mplib figures thereafter in the scope. And the options listed above are provided for `\mpfig` and `\usemplibgroup` (see [below § 1.2.15](#)) commands as well.

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

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

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

\mplibgroup{...}          % see below
  \mpfig[off]             % do not tag this figure
  ...
  \endmpfig
\endmplibgroup

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

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

1.2 METAPOST

1.2.1 T_EX dimen and color

`mplibdimen  $\langle$ string $\rangle$` and `mplibcolor  $\langle$ string $\rangle$` are METAPOST interfaces for the T_EX 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 T_EX commands outside of the `btex` or `verbatimtex ... etex`.

1.2.2 More on T_EX color

`mplibtexcolor  $\langle$ string $\rangle$` is a METAPOST operator that converts a T_EX color expression to a METAPOST color expression, that can be used anywhere color expression is expected as well as after

the `withcolor` command.⁹ For instance:

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

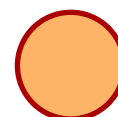
But the result may vary in its color model (gray/rgb/cmyk) according to the given $\TeX$ color. Therefore the example shown above would raise a METAPOST error: `cmykcolor col;` should have been declared. By contrast, `mplibrbgtexcolor` $\langle string \rangle$ always returns rgb-model expressions.

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

1.2.3 Fill and stroke colors

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` $(\langle fill\ color\ expr \rangle, \langle stroke\ color\ expr \rangle)$. 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 § 1.2.8):

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

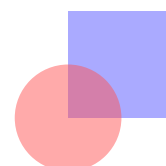


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 Transparency

`withtransparency` $(\langle number \rangle | \langle string \rangle, \langle numeric \rangle)$ is provided for *plain* format as well as *metafun*. The first argument accepts a number or a name among alternative transparency methods (a.k.a. *blend modes*; 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
```

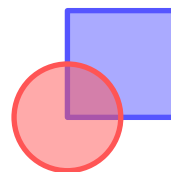


⁹Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color.

1.2.5 Fill and stroke transparency

By analogy with the macro `withmplibcolors` (see above § 1.2.3), the macro `withmplibopacities` is also provided. The syntax is `withmplibopacities (<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]
  withmplibopacities (1, 1/2, 1)    % or ("normal", 1/2, 1)
;
filldraw fullcircle scaled 40
  withcolor 1/3[red,white]
  withmplibopacities (1, 1/2, 1)
;
\endmpfig
```



1.2.6 Text outline as a text image

`mplibgraphicstext <string>` is a METAPOST operator, the effect of which is similar to that of ConT_EXt's `graphicstext` or our own `mpliboutlinetext` (see below § 1.2.9). However the syntax is somewhat different.

```
draw mplibgraphicstext "$\sqrt{2+x}$"
  rotated 10 scaled 3
  fakebold 2.5                % fontspec option
  fillcolor "red!50"          % color expression
  drawcolor 2/3 red           % 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 `mplibgraphicstext` 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, `mplibgraphicstext` will produce better results than ConT_EXt or even than our own `mpliboutlinetext`, 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 `mpliboutlinetext`, you cannot manipulate the shape of outline paths, because the returned picture is basically a `btex ... etex` picture.

¹⁰Users can use the `withmplibcolors` 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 below § 1.2.12.

1.2.7 Glyph outline

METAPOST operator `mplibglyph` $\langle number \rangle$ | $\langle string \rangle$ of $\langle number \rangle$ | $\langle string \rangle$ 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.

```
mplibglyph 50 of \fontid\font          % slot 50 of current font
mplibglyph "Q" of "TU/TeXGyrePagella(0)/m/n/10" % font csname
mplibglyph "Q" of "texgyrepagella-regular.otf"   % raw filename
mplibglyph "R" of "utmr8a.pfb"                  % raw filename (type1 font)
mplibglyph "Q" of "Times.ttc(2)"                % subfont number
mplibglyph "Q" of "SourceHanSansK-VF.otf[Regular]" % instance name
mplibglyph "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 TeX font csname (without backslash) or the raw filename of a font. When it is a font filename, a number within parentheses after the filename denotes a subfont number (starting from zero) of a TTC font; a string within brackets denotes an instance name, 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 `mplibglyph`.

1.2.8 Drawing a glyph outline

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` $\langle picture \rangle$ command fills the paths according to the nonzero winding number rule. As a result, for instance, the area surrounded by inner path of “O” will remain transparent.

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` $\langle the\ last\ path \rangle$ `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` $\langle picture \rangle$, `mplibstrokeglyph` $\langle picture \rangle$, and `mplibfillglyph` $\langle picture \rangle$, the last one being a synonym of `mplibdrawglyph` command.

¹²The bug seems to be fixed in `font-cff.lmt` contained in ConTeXt mkxl, but current `luaotfload` is based on `font-cff.lua` from ConTeXt mkiv. As you see, `mplibglyph` operator requires `luaotfload` package loaded, which however is done automatically by $\mathcal{E}\mathcal{T}\mathcal{E}\mathcal{X}$ format.

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

An example:

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



1.2.9 Text outline as a path image

As said before at § 1.1.3, `luamplib` provides the METAPOST operator `mpliboutlinetext` ($\langle string \rangle$) 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.

1.2.10 Unicode-aware length

`mpliblength` ($\langle string \rangle$) 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` ($\langle string \rangle$) 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.

1.2.11 Unicode-aware substring

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

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

1.2.12 Shading

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`, `mplibgraphicstext` (see below § 1.2.6), 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 painted *without* color can also be regarded as a textual picture.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.
- A few more optional macros are available in addition to the *metafun*'s: `withshadingpoints`, `withshadingcenters`, `withshadingextend`, `withshadingmatrix`, and `withshadingstroke`.
- More shading methods are available: "triangle", "lattice", "coons", and "tensor". See below § 1.2.18, § 1.2.19, § 1.2.20, and § 1.2.21 for these methods.

The syntax is `<path> | <textual picture> withshadingmethod <string>`, where the latter shall be "linear", "circular", "triangle", "lattice", "coons", or "tensor". 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.

¹⁴See the last [example](#) in this subsection. See also below § 1.2.8, particularly the first [example](#) of tiling pattern at § 1.2.14. See also § 1.2.15 and § 1.2.16, particularly the example in the [note](#) about picture shading with transparency group.

withshadingdirection $\langle pair \rangle$ Starting and ending points (as time value) on the bounding box, default value being $(0,2)$.

withshadingorigin $\langle pair \rangle$ The center of both starting and ending circles, default value being center p , where p is the operand of `withshadingmethod`.

withshadingcenter $\langle pair \rangle$ 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 decalred.

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 $\text{\TeX}$ 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.

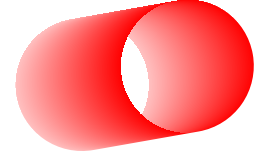
withshadingextend ($\langle boolean \rangle$, $\langle boolean \rangle$) Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

`\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

```



withshadingmatrix $\langle \text{string} \rangle$ METAPOST code for transformation of shading, such as "xscaled 1.2 yscaled 0.8"; or six numerics separated by spaces, such as "1.2 0 0 0.8 0 0" (xx, yx, xy, yy, x, and y-part respectively).

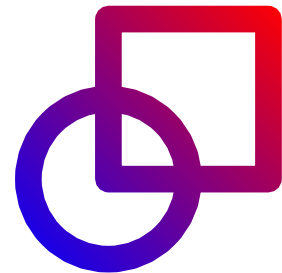
withshadingstroke $\langle \text{string} \rangle$ When the shading object is a $\langle \text{path} \rangle$, the string shall be "yes" or "no": if "yes", we get the path stroked and *then* shaded. It could be more efficient than issuing two sentences.

When the shading object is a $\langle \text{picture} \rangle$, the string shall be "filldraw", "both", "draw", "yes", or "no": "filldraw" or "both" will shade both the fill part and the stroke part; "draw" or "yes" will shade the stroke part only; all other cases will shade the fill part only, which is the default behavior. An example:

```

\mpfig
pickup pencircle scaled 10;
draw image(
    draw fullcircle scaled 60 withoutcolor;
    draw unitsquare scaled 60 withoutcolor;
)
withshadingmethod "linear"
withshadingcolors (blue, red)
withshadingstroke "yes" ;
\endmpfig

```



1.2.13 Fading

METAPOST command **withfademethod** makes the color of an object gradiently transparent, a.k.a. *fading*. The syntax is $\langle \text{path} \rangle$ | $\langle \text{picture} \rangle$ **withfademethod** $\langle \text{string} \rangle$, 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 § 1.2.17.

Related macros to control optional values are:

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

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

withfadecenter is a synonym of withfadevector.

withfaderadius (*<numeric>*, *<numeric>*) 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.

withfadestep (...) for combined fading of more than two opacities.

withfadefraction *<numeric>* Fractional number of each fading step. Only meaningful within withfadestep.

withfadeextend (*<boolean>*, *<boolean>*) specifies whether to extend the fading beyond the starting and ending points or circles, default value being (true, true).

withfadematrix *<string>* METAPOST code for transformation of fading, such as "xscaled 1.2 yscaled 0.8"; or six numerics separated by spaces, such as "1.2 0 0 0.8 0 0" (xx, yx, xy, yy, x, and y-part respectively).

withfadebbox (*<pair>*, *<pair>*) sets the bounding box of the fading area, default value being (llcorner p, urcorner p). Though this option is not needed in most cases, there could be cases when users want to explicitly control the bounding box. Particularly, see the description [below](#) at § 1.2.15 on the analogous macro withgroupbbox.

An example:

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



1.2.14 Tiling pattern

T_EX 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.¹⁵ As said before at § 1.2.12, the *textual picture* here means basically any text typeset by T_EX, mostly the result of the `btex` command (and its derivatives) or the `infont` operator.

¹⁵`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

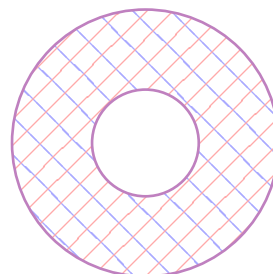
An example:

```

\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.

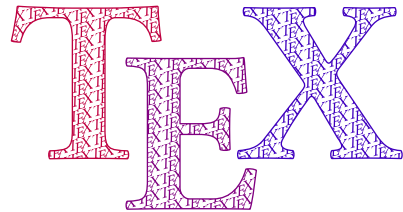
When you use special effect such as transparency in a pattern cell, *resources* option is

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.

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).¹⁶ 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"
    fakebold 1/2
    rotated 10 scaled 8
    withmppattern "pattnocolor"
    withmplibcolors (
      2/3[red,white],      % paints the pattern
      2/3 red
    ) ;
endfig;
\end{mplibcode}
```

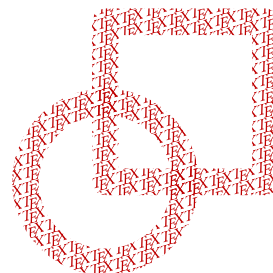


After v2.42.6, we provide an optional macro for both colored and uncolored patterns:

¹⁶When using DVI mode, `-c` option might be needed to the `dvipdfmx` command.

withpatternstroke $\langle string \rangle$ where the string shall be "filldraw", "both", "draw", "yes", or "no": "filldraw" or "both" will tile both the fill part and the stroke part; "draw" or "yes" will tile the stroke part only; all other cases will tile the fill part only, which is the default behavior. An example:

```
\mpfig
pickup pencircle scaled 10;
draw image(
  draw fullcircle scaled 60
    withcolor 3/4 red ;      % paints the pattern
  draw unitsquare scaled 60
    withoutcolor ;
)
withmppattern "pattnocolor"
withpatternstroke "yes" ;
\endmpfig
```



1.2.15 FormXObject from METAPOST side

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: $\langle picture \rangle$ | $\langle path \rangle$ *asgroup* $\langle string \rangle$, the latter being "" (empty string) or any comma-separated combination of isolated, knockout, wrapped, and off (for example, "isolated,knockout,wrapped"), which will return a METAPOST picture. The additional features provided by *luamplib* are:

- As mentioned, in addition to those arguments mimicking *metafun*'s, we allow other optional arguments at the right-hand side:
 - *asgroup* "off" will produce an ordinary *form XObject* rather than a transparency group XObject. By contrast, a transparency group will be produced when 'off' is not given, including "" (empty string) in which case both of the PDF keys /I and /K are false.
 - *asgroup* "wrapped" will produce a transparency group XObject in which an ordinary form XObject is wrapped up. This option could be useful when a picture shading (*shading pattern* in technical terms) is used in the object of *asgroup*. See the note about picture shading described [below](#).
- You can reuse the XObject as many times as you want in the $\text{\TeX}$ code or in other METAPOST code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide $\text{\TeX}$ and METAPOST macros as follows:

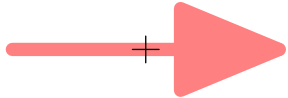
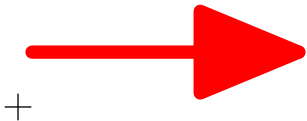
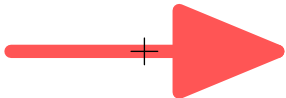
withgroupname $\langle string \rangle$ associates an XObject with the given name. When this command is not appended to the sentence with *asgroup* operator, the default name 'lastmplibgroup' will be used.

`\usemplibgroup{⟨name⟩}` is a T_EX command to reuse an XObject of the name once used. Note that the position of the XObject will be origin-based: in other words, lower-left corner of the bounding box will be shifted to the origin.

`usemplibgroup ⟨string⟩` is a METAPOST command which will add an XObject of the name to the currentpicture. Contrary to the T_EX command just mentioned, the position of the XObject is the same as the original XObject.

`withgroupbbox (⟨pair⟩, ⟨pair⟩)` sets the bounding box of the XObject, 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. However, this option is not needed in most cases as luamplib v2.42.9 or after tries to detect pen width information.

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

<pre> \mpfig picture pic; pic = image(drawarrow (left--right) scaled 5 withcolor red) scaled 10 ; draw pic asgroup "off" withtransparency (1, 1/2) ; \endmpfig </pre>	
<pre> \mpfig draw pic asgroup "" withgroupname "mygroup" withtransparency (1, 1/2) ; draw (left--right) scaled 5 ; draw (up--down) scaled 5 ; \endmpfig </pre>	
<pre> \noindent \clap{\vrule width 10bp height .25bp depth .25bp}% \clap{\vrule width .5bp height 5bp depth 5bp}% \usemplibgroup{mygroup} </pre>	
<pre> \mpfig usemplibgroup "mygroup" withtransparency (1, 2/3) ; draw (left--right) scaled 5 ; draw (up--down) scaled 5 ; \endmpfig </pre>	

Also note that normally the XObjects are not affected by outer color commands. However, if you have made the original XObject using `withoutcolor` command, colors will have effects on its uncolored objects.

Note on picture shading When you give shading effect upon a *textual picture* (i.e. 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 or MuPDF. 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 up in a transparency group XObject. This sort of double wrapping will be done automatically when you specify 'wrapped' as an optional argument of asgroup. Most PDF renderers do render it properly:

```
\mpfig
draw pic2
asgroup "wrapped"
withtransparency (1, 2/3) ;
\endmpfig
```



or preferably (see next subsection § 1.2.16 on \mplibgroup):

```
\mplibgroup{myshading}[asgroup="wrapped"]
```

```

\mpfig
  draw pic2 ;
\endmpfig
\endmplibgroup

```

```

\mpfig
  usemplibgroup "myshading" rotated 15
  withtransparency (1, 2/3) ;
\endmpfig

```



1.2.16 Form XObject from T_EX side

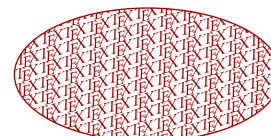
T_EX macros `\mplibgroup{<name>}` ... `\endmplibgroup` are described here in this subsection, as they are deeply related to the `asgroup` operator described just above at § 1.2.15. With these, users can define a transparency group or an ordinary *form XObject* from T_EX side. The syntax is similar to the `\mppattern` command described above at § 1.2.14.

An example:¹⁷

```

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

```



```

\usemplibgroup{texpatt}

\mpfig
  usemplibgroup "texpatt"
  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 § 1.2.15.

¹⁷Note that, here again by the option `asgroup="wrapped"`, within a transparency group XObject a tiling pattern is wrapped up 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.

Table 2: options for \mplibgroup

Key	Value Type	Explanation
asgroup	<i>string</i>	"" , or any comma-separated combination of isolated, knockout, wrapped, and off; or "masking"
colorspace	<i>string</i>	/CS entry to a transparency group, such as "/DeviceGray", "/DeviceRGB", or "/DeviceCMYK"
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

Option asgroup="wrapped" can be regarded as a shortcut of double \mplibgroup command. The content will be wrapped up in an ordinary formXObject before being wrapped again in a transparency group. This option could be useful when a tiling pattern (see the example just above) or a picture shading (see the [example](#) above at § 1.2.15) is used in the content.

As for the option asgroup="masking", see the next subsection § 1.2.17.

With colorspace option, which is no-op for ordinary formXObject, 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.¹⁸

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 § 1.2.15, excepting that the mplibgroup made by T_EX code (not by METAPOST code) respects original height and depth.

1.2.17 Masking

Using the command withmaskinggroup, the mplibgroup (see above § 1.2.16) 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.¹⁹

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.²⁰

¹⁸Note that some PDF renderers such as Mac OS Preview or Firefox do not render properly this option.

¹⁹In 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 § 1.2.16).

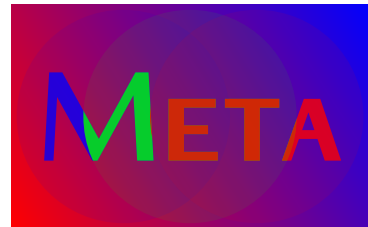
²⁰Color expressions in rgb or cmyk model are also allowed, in accordance with the option colorspace="/DeviceRGB"

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

\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
```



1.2.18 Free-form Gouraud-shaded triangle mesh shading

The syntax of this type of shading is $\langle path \rangle$ | $\langle textual picture \rangle$ `withshadingmethod "triangle"`, as the area to be shaded is defined by a series of triangles. Among a number of optional macros for shading, only `withshadingmatrix` and `withshadingstroke` are available (see above § 1.2.12).

Optional macros for triangle mesh shading:

withtrianglepatchinit ($\langle path \rangle$ | $\langle string \rangle$, $\langle color \rangle$, $\langle color \rangle$, $\langle color \rangle$) The initial triangle. This macro can be used multiple times.

The first argument shall be a path that has three (or more) vertices, or a string composed of six numerics separated by spaces (x and y coordinates at each point). When this macro is not given, the default path value will be the object of shading.

Remaining arguments are three colors for each vertex in the order of the points. When this macro is not given, the default color values will be red, green, and blue.

withtrianglepatchnext ($\langle number \rangle$, $\langle pair \rangle$ | $\langle string \rangle$, $\langle color \rangle$) The new triangle attached to the previous one. This optional macro requires `withtrianglepatchinit` specified beforehand.

or `colorspace="/DeviceCMYK"` given to the masking transparency group. This however we do not recommend as the luminosity is difficult to understand intuitively.

The first argument shall be a number (1 or 2) denoting the edge to which a new triangle will be attached. Edge 1 is the last explicit segment of the previous triangle; edge 2 is the implicit segment of the previous triangle.

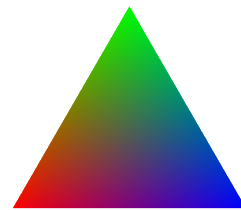
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates).

The third argument is the color for the new vertex.

Examples showing the triangle shading and illustrating the usage of the optional macros:

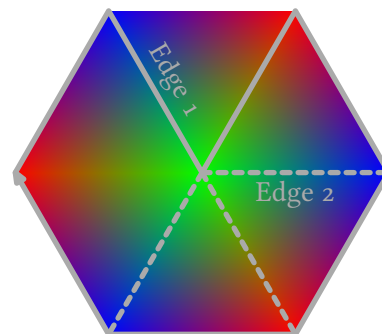
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

q := origin ;
draw ( q -- qadd(60) -- qadd(-60) ) scaled 5/4
  withshadingmethod "triangle" ;
\endmpfig
```



```
\mpfig
q := origin ;
path p ;
p = q -- qadd(60) -- qadd(-60) -- qadd(60) --
  qadd(-60) -- qadd(-120) -- qadd(-180) -- cycle ;

draw bbox p
  withshadingmethod "triangle"
  withtrianglepatchinit (p, red, blue, green)
  withtrianglepatchnext (1, point 3 of p, red)
  withtrianglepatchnext (1, point 4 of p, blue)
  withtrianglepatchnext (2, point 5 of p, red)
  withtrianglepatchnext (2, point 6 of p, blue)
  withtrianglepatchnext (2, point 0 of p, red) ;
\endmpfig
```



1.2.19 Lattice-form Gouraud-shaded triangle mesh shading

This type of shading is similar to the triangle mesh shading described just above, but the vertices are organized into rows, which need not be geometrically linear. The syntax is $\langle path \rangle | \langle textual picture \rangle$ withshadingmethod "lattice". Among a number of optional macros for shading, only withshadingmatrix and withshadingstroke are available (see above § 1.2.12).

Optional macros for lattice-form shading:

withlatticeverticesperrow $\langle number \rangle$ The number of vertices in each row of the lattice. The value should be greater than or equal to 2. The default value is 2.

withlatticeverticesdata (*<pair>* | *<string>*, *<color>*, ...) A list of vertices and colors.

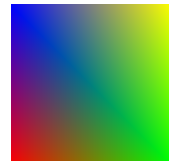
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

This combination of a point and a color should be repeated multiple times, which must be a multiple of the number of vertices in each row.

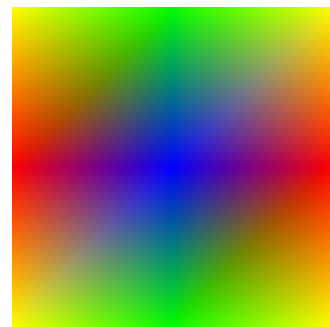
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

Examples showing the lattice-form shading and illustrating the usage of the optional macros:

```
\mpfig
  u := 60;
  draw unitsquare scaled u
    withshadingmethod "lattice" ;
\endmpfig
```



```
\mpfig
  color yellow;
  yellow = (1,1,0);
  draw unitsquare scaled 2u
    withshadingmethod "lattice"
    withlatticeverticesperrow 3
    withlatticeverticesdata (
      (0,2u),yellow, (u,2u),green, (2u,2u),yellow,
      (0, u),red , (u, u),blue , (2u, u),red ,
      (0, 0),yellow, (u, 0),green, (2u, 0),yellow,
    ) ;
\endmpfig
```



1.2.20 Coons patch mesh shading

Coons patch mesh shadings are constructed from one or more color patches, each bounded by four cubic Bézier curves. The syntax is *<path>* | *<textual picture>* *withshadingmethod "coons"*. Among a number of optional macros for shading, only *withshadingmatrix* and *withshadingstroke* are available (see above § 1.2.12).

Optional macros for Coons patch mesh shading:

withcoonspatchinit (*<path>* | *<string>*, *<color>*, *<color>*, *<color>*, *<color>*) The initial patch. This macro can be used multiple times.

The first argument shall be a closed path that has four vertices, or a string composed of twenty-four numerics separated by spaces (xpart point 0 of p, ypart point 0 of p, ..., xpart precontrol 0 of p, ypart precontrol 0 of p). When this macro is not given, the default path value will be obtained from the object of shading.

Remaining arguments are four colors for each vertex in the order of the points. When this macro is not given, the default color values will be red, green, blue, and yellow.

withcoonspatchnext ($\langle \text{number} \rangle$, $\langle \text{path} \rangle$ | $\langle \text{string} \rangle$, $\langle \text{color} \rangle$, $\langle \text{color} \rangle$) The new patch attached to the previous one. This optional macro requires `withcoonspatchinit` specified beforehand.

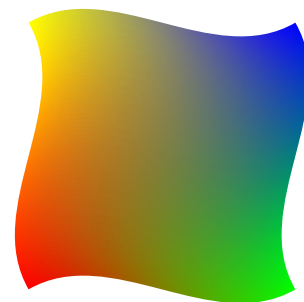
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

The second argument shall be a path that has four vertices, or a string of sixteen numerics separated by spaces (xpart postcontrol 1 of p, ypart postcontrol 1 of p, ..., xpart precontrol 0 of p, ypart precontrol 0 of p). The orientation of the new path should be reversed from the previous one, and it is assumed that the first segment of the new path coincides with the edge segment just mentioned.

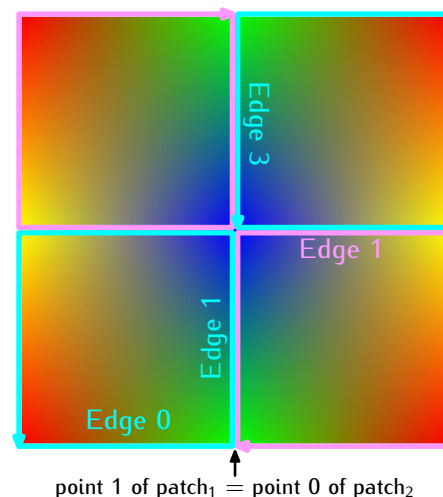
Remaining arguments are two color expressions for the new vertices.

Examples showing the Coons patch shading and illustrating the usage of the optional macros:

```
\mpfig
draw (
  (0,0) {dir 30} .. {dir 30}
  (100,0) {dir 120} .. {dir 120}
  (100,100) {dir 210} .. {dir 210}
  (0,100) {dir 300} .. {dir 300}
  cycle
)
withshadingmethod "coons" ;
\endmpfig
```



```
\mpfig
u := 80 ;
path p;
p = unitsquare scaled u;
def fsquare (expr n) =
  ( point n of p --
    point n+1 of p --
    point n+2 of p --
    point n+3 of p --
    cycle )
enddef;
def rsquare (expr n) =
  ( point n of p --
    point n-1 of p --
    point n-2 of p --
    point n-3 of p --
    cycle )
enddef;
fill p scaled 2
  withshadingmethod "coons"
  withcoonspatchinit
```



```

    (fsquare(0), red, green, blue, yellow)
  withcoonspatchnext
    (1, rsquare(0) shifted (u,0), yellow, red)
  withcoonspatchnext
    (1, fsquare(0) shifted (u,u), red, green)
  withcoonspatchnext
    (3, rsquare(2) shifted (0,u), yellow, red) ;
\endmpfig

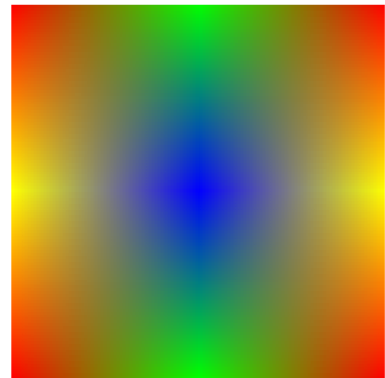
```

N.B. Be aware that currently Mac OS Preview exposes its some bug in rendering the figure just above, especially when the edge flag is 3. In order to circumvent the Preview's bug, you can use withcoonspatchinit multiple times:

```

\mpfig
  u := 70 ;
  p := unitsquare scaled u ;
  fill p scaled 2
    withshadingmethod "coons"
    withcoonspatchinit
      (p, red, green, blue, yellow)
    withcoonspatchinit
      (p shifted (u,0), green, red, yellow, blue)
    withcoonspatchinit
      (p shifted (u,u), blue, yellow, red, green)
    withcoonspatchinit
      (p shifted (0,u), yellow, blue, green, red) ;
\endmpfig

```



1.2.21 Tensor-product patch mesh shading

This type is almost identical to the Coons patch mesh shading, except that tensor-product patch has four additional, *internal* control points to adjust the color mapping. The syntax is $\langle path \rangle$ | $\langle textual\ picture \rangle$ withshadingmethod "tensor". Among a number of optional macros for shading, only withshadingmatrix and withshadingstroke are available (see above § 1.2.12).

Optional macros for tensor-product patch mesh shading:

withtensorpatchinit ($\langle path \rangle$ | $\langle string \rangle$, $\langle path \rangle$ | $\langle string \rangle$, $\langle color \rangle$, $\langle color \rangle$, $\langle color \rangle$, $\langle color \rangle$) The initial patch. This macro can be used multiple times.

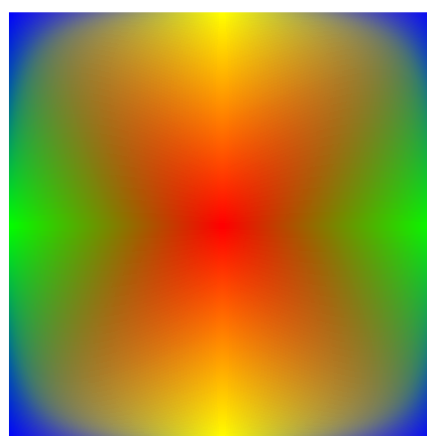
The only difference to withcoonspatchinit macro of Coons patch shading is the second argument inserted for specifying four inner control points. It shall be a path that has four points, or a string composed of eight numerics separated by spaces (x and y coordinates of each point). If the first argument is given in string type, the second argument should also be given in string type. It is assumed that the orientation of the inner path is the same as the outer path. When this macro is not given, the default value will be an imaginary path scaled by half the size of the outer path.

withtensorpatchnext (*⟨number⟩*, *⟨path⟩* | *⟨string⟩*, *⟨path⟩* | *⟨string⟩*, *⟨color⟩*, *⟨color⟩*) The new patch attached to the previous one. This optional macro requires `withtensorpatchinit` specified beforehand.

The only difference to `withcoonspatchnext` macro of Coons patch shading is the third argument inserted for specifying four inner control points. The syntax is the same as the second argument of `withtensorpatchinit` described just above.

An example to show the effect of tensor-product patch mesh shading:

```
\mpfig
u := 80 ;
path p, q ;
p = unitsquare scaled u ;
q = p scaled 1/10 shifted (dir 45 * 1.2u) ;
fill p scaled 2 shifted (-u,-u)
  withshadingmethod "tensor"
  withtensorpatchinit
    (p, q, red, green, blue, yellow)
  withtensorpatchinit
    (p rotated 90, q rotated 90,
     red, yellow, blue, green)
  withtensorpatchinit
    (p rotated 180, q rotated 180,
     red, green, blue, yellow)
  withtensorpatchinit
    (p rotated 270, q rotated 270,
     red, yellow, blue, green) ;
\endmpfig
```



N.B. Be aware that currently Mac OS Preview or Foxit Editor does not render properly the figure above.

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 Accessing METAPOST variables

Users can access the Lua table containing mplib instances, `luampbib.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 = luampbib.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(luampbib.instances["#1"]:get_numeric"#2"))
}}
\mpnumeric{myinstance}{n}\relax
```

3.0

1.3.3 Running a METAPOST code

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

```
luampbib.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 `luampbib` namespace, listed in Table 3, can affect the process of `process_mplibcode`.

1.3.4 Registering a tiling pattern

This is the Lua interface for `\mppattern ... \endmppattern` described above at § 1.2.14.

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

Table 3: elements in luamplib table (partial)

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

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 0.²¹

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 Registering a formXObject

This is the Lua interface for `\mplibgroup ... \endmplibgroup` described above at § 1.2.16.

```
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 T_EX (not METAPOST) code, please make sure that both of the T_EX macros ‘`MPllx`’ and ‘`MPlly`’ are defined as ‘`0`’ before invoking the Lua function.²²

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 T_EX macro ‘`luamplib.group.<name>`’. If you need the `boxresource` index, inspect this macro using `token.get_macro` function.

2 Implementation

2.1 Lua module

²¹In DVI mode, T_EX macro ‘`mplibpatternname`’ should be set as $\langle pattern name \rangle$ before preparing the box, if shading pattern (i.e. shading on picture) is used in the pattern cell.

²²In DVI mode, T_EX macro ‘`mplibgroupname`’ also should be set as $\langle group name \rangle$ before preparing the box, if shading pattern (i.e. shading on picture) is used in the `mplibgroup`.

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version    = "2.42.9",
5   date      = "2026/09/01",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8

```

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

```

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

```

Use our own function for warn/info/err.

```

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

```

```

46 end
47
48 luamplib.showlog = luamplib.showlog or false
49

```

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 ioopen        = 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 .. "/_luamplib_temp_file_"
78     local fh = ioopen(name,"w")
79     if fh then
80       fh:close(); os.remove(name)
81       return true
82     end
83   end
84 end
85 local mk_full_path = lfs.mkdirp or lfs.mkdirs or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\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

```

We have to replace `btex ... etex` and `verbatimtex ... etex` in input files, if needed. A plain `gsub` will not do, as the scanner of `METAPOST` itself skips comments and string literals: a `btex` occurring in them is just a word, and pairing it with the next real `etex` would silently swallow the code in between (issue #204). So we scan the code the way `METAPOST` does, handing each block, string and comment over to the callbacks in action: `btex` and `verbatimtex` get the body of a block, `str` the contents of a string literal, and `comment` the comment including its percent sign. A missing callback means *leave it alone*. Inside a block, on the other hand, nothing but the next `etex` matters—comments are not special there, exactly as in `METAPOST`. Finally, `escapepercent` tells that a percent sign preceded by a backslash is not the start of a comment.

```

147 local name_b = "%f[%a_]"
148 local name_e = "%f[^%a_]"
149 local etex_pat = name_b.."etex"..name_e
150 local special_pat = "[%%"%a_]' -- start of a comment, a string, or a name
151
152 local function trim (str)
153   return (str:gsub("^%s+", ""):gsub("%s+$", ""))
154 end
155
156 local function scan_mpcode (data, action)
157   local res, n, pos, i, count = {}, 0, 1, 1, 0
158   local len = #data
159   while i <= len do
160     local s = data:find(special_pat, i)
161     if not s then break end
162     local c = data:sub(s,s)
163     if c == "%" then -- a comment runs to the end of the line
164       if action.escapepercent and data:sub(s-1,s-1) == "\\" then
165         i = s + 1
166       else
167         local nl = data:find("\n", s, true) or len + 1
168         if action.comment then
169           res[n+1], res[n+2] = data:sub(pos,s-1), action.comment(data:sub(s,nl-1))
170           n, pos = n+2, nl
171         end
172         i = nl
173       end
174     elseif c == "'" then -- a string never spans a line

```

```

175     local q = data:find('[\n]', s+1)
176     if q and data:sub(q,q) == ''' then
177         if action.str then
178             res[n+1], res[n+2] = data:sub(pos,s-1), action.str(data:sub(s+1,q-1))
179             n, pos = n+2, q+1
180         end
181         i = q + 1
182     else
183         i = q or len + 1 -- unterminated: MetaPost flushes it at the line end
184     end
185 else
186     local _, e = data:find("^[%a_]+", s)
187     local block = data:sub(s,e)
188     block = block == "btex" and action.btex
189         or block == "verbatimtex" and action.verbatimtex
190     local es, ee
191     if block then es, ee = data:find(etex_pat, e+1) end
192     if es then
193         res[n+1], res[n+2] = data:sub(pos,s-1), block(trim(data:sub(e+1,es-1)))
194         n, pos, i, count = n+2, ee+1, ee+1, count+1
195     else
196         i = e + 1 -- no etex in sight: not a block after all
197     end
198 end
199 end
200 if pos == 1 then return data, count end
201 res[n+1] = data:sub(pos)
202 return tableconcat(res), count
203 end
204
205 local replace_texblock = {
206     btex      = function(str) return format("btex %s etex ", str) end, -- space
207     verbatimtex = function(str) return format("verbatimtex %s etex;", str) end, -- semicolon
208 }
209

```

Function `luamplib.finder`

```

210 local currenttime = os.time()
211 do
212     local luamplibtime = lfsattributes(kpse.find_file"luamplib.lua", "modification")

```

`format.mp` is much complicated, so specially treated.

```

213 local function replaceformatmp(file,newfile,ofmodify)
214     local fh = ioopen(file,"r")
215     if not fh then return file end
216     local data = fh:read("*all"); fh:close()
217     fh = ioopen(newfile,"w")
218     if not fh then return file end
219     fh:write(
220         "let normalinfont = infont;\n",

```

```

221     "primarydef str infont name = rawtexttext(str) enddef;\n",
222     data,
223     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
224     "vardef Fexp_(expr x) = rawtexttext(\"$^{\"&decimal x&\"}$\") enddef;\n",
225     "let infont = normalinfont;\n"
226 ); fh:close()
227 lfstouch(newfile,currenttime,ofmodify)
228 return newfile
229 end
230 local function replaceinputmpfile (name,file)
231     local ofmodify = lfsattributes(file,"modification")
232     if not ofmodify then return file end
233     local newfile = name:gsub("%W","_")
234     newfile = format("%s/luamplib_input_%s", cachedir or outputdir(), newfile)
235     if newfile and luamplibtime then
236         local nf = lfsattributes(newfile)
237         if nf and nf.mode == "file" and
238             ofmodify == nf.modification and luamplibtime < nf.access then
239             return nf.size == 0 and file or newfile
240         end
241     end
242     if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
243     local fh = ioopen(file,"r")
244     if not fh then return file end
245     local data = fh:read("*all"); fh:close()

```

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

```

246     local count
247     data, count = scan_mpcode(data, replace_tblock)
248     if count == 0 then
249         needtoreplace[name] = true
250         fh = ioopen(newfile,"w");
251         if fh then
252             fh:close()
253             lfstouch(newfile,currenttime,ofmodify)
254         end
255         return file
256     end
257     fh = ioopen(newfile,"w")
258     if not fh then return file end
259     fh:write(data); fh:close()
260     lfstouch(newfile,currenttime,ofmodify)
261     return newfile
262 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.

```

263 local mpkpse
264 do

```

```

265     local exe = 0
266     while arg[exe-1] do
267         exe = exe-1
268     end
269     mpkpse = kpse.new(arg[exe], "mpost")
270 end
271 local special_ftype = {
272     pfb = "type1 fonts",
273     enc = "enc files",
274 }
275 function luamplib.finder (name, mode, ftype)
276     if mode == "w" then
277         if name and name ~= "mpout.log" then
278             kpse.record_output_file(name) -- recorder
279         end
280         return name
281     else
282         ftype = special_ftype[ftype] or ftype
283         local file = mpkpse.find_file(name, ftype)
284         if file then
285             if lfstouch and ftype == "mp" and not noneedtoreplace[name] and not noneedtoreplace["*.mp"] then
286                 file = replaceinputmpfile(name, file)
287             end
288         else
289             file = mpkpse.find_file(name, name:match("%a+$"))
290         end
291         if file then
292             kpse.record_input_file(file) -- recorder
293         end
294         return file
295     end
296 end
297 end
298

```

For the main function: process

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

```

299 local currentformat = "plain"
300 function luamplib.setformat (name)
301     currentformat = name
302 end

```

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

```

303 luamplib.codeinherit = false
304 local mplibinstances = {}
305 luamplib.instances = mplibinstances
306 local has_instancename = false
307
308 local process
309 do

```

```

310 local function reporterror (result, prevlog)
311   if not result then
312     err("no result object returned")
313   else
314     local t, e, l = result.term, result.error, result.log
log has more information than term, so log first (2021/08/02)
315     local log = l or t or "no-term"
316     log = log:gsub("%(Please type a command or say `end`)", ""):gsub("\n+", "\n")
317     if result.status > 0 then
318       local first = log:match"(.-\n! .-)\n! "
319       if first then
320         termorlog("term", first)
321         termorlog("log", log, "Warning")
322       else
323         warn(log)
324       end
325       if result.status > 1 then
326         err(e or "see above messages")
327       end
328     elseif prevlog then
329       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.

```

330     local show = log:match"\n>>? .+"
331     if show then
332       termorlog("term", show, "Info (more info in the log)")
333       info(log)
334     elseif luamplib.showlog and log:find"%g" then
335       info(log)
336     end
337   end
338   return log
339 end
340 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.

```

341 if not math.initialseed then math.randomseed(currenttime) end
342 local function luamplibload (name)
343   local mpx = mplib.new {
344     ini_version = true,
345     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>.

```

346   make_text   = luamplib.maketext,
347   run_script  = luamplib.runscript,
348   math_mode   = luamplib.numbersystem,
349   job_name    = tex.jobname,

```

```

350     random_seed = math.random(4095),
351     utf8_mode   = true,
352     extensions  = 1,
353 }

```

Append our own METAPOST preamble to the preamble loading plain/metafun format.

```

354 local preamble = tableconcat{
355     format(luamplib.preambles.preamble, replacesuffix(name,"mp")),
356     luamplib.preambles.mplibcode,
357     luamplib.legacyverbatim and luamplib.preambles.legacyverbatim or "",
358     luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
359 }
360 local result, log
361 if not mpx then
362     result = { status = 99, error = "out of memory"}
363 else
364     result = mpx:execute(preamble)
365 end
366 log = reporterror(result)
367 return mpx, result, log
368 end

```

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

```

369 local process_stack = 0
370 function process (data, instancename)
371     local currfmt
372     process_stack = process_stack + 1
373     if instancename and instancename ~= "" then
374         currfmt = instancename
375         has_instancename = true
376     else
377         currfmt = tableconcat{
378             currentformat,
379             luamplib.numbersystem or "scaled",
380             tostring(luamplib.texttextlabel),
381             tostring(luamplib.legacyverbatim),
382             tostring(process_stack), -- try to address #63
383         }
384         has_instancename = false
385     end
386     local mpx = mplibinstances[currfmt]
387     local standalone = not (has_instancename or luamplib.codeinherit)
388     if mpx and standalone then
389         mpx:finish()
390     end
391     local log = ""
392     if standalone or not mpx then
393         mpx, _, log = luamplibload(currentformat)
394         mplibinstances[currfmt] = mpx
395     end

```

```

396 local converted, result = false, {}
397 if mpx and data then
398   result = mpx:execute(data)
399   local log = reporterror(result, log)
400   if log then
401     if result.fig then
402       converted = luamplib.convert(result)
403     end
404   end
405 else
406   err"Mem file unloadable. Maybe generated with a different version of mplib?"
407 end
408 process_stack = process_stack - 1
409 return converted, result
410 end
411 end
412

```

dvipdfmx is supported, though nobody seems to use it.

```

413 local pdfmode = tex.outputmode > 0
414

```

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

```

415 local catlatex = luatexbase.registernumber("catcodetable@latex")
416 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.

```

417 local function run_tex_code (str, cat)
418   texruntoks(function() texsprint(cat or catlatex, str) end)
419 end

```

For conversion of sp to bp.

```

420 local factor = 65536*(7227/7200)
421

```

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.

```

422 local texboxes = { globalid = 0, localid = 4096 }
423 local process_tex_text
424 do
425   local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
426     xscaled %f yscaled %f shifted (0,-%f) \z
427     withprescript "mplibtexboxid=%i:%f:%f")'
428   function process_tex_text (str, maketext)
429     if str then
430       if not maketext then str = str:gsub("\r.-$", "") end
431       local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
432         and "\\global" or ""

```

```

433     local tex_box_id
434     if global == "" then
435         tex_box_id = texboxes.localid + 1
436         texboxes.localid = tex_box_id
437     else
438         local boxid = texboxes.globalid + 1
439         texboxes.globalid = boxid
440         run_tex_code(format([[\\expandafter\\newbox\\csname luamplib.box.\\s\\endcsname]], boxid))
441         tex_box_id = tex.getcount'allocationnumber'
442     end
443     if str:find"^[taggingoff%]" then
444         str = str:gsub("^[taggingoff%]%s*", "")
445         run_tex_code(format("\\luamplibnotagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
446                             tex_box_id, global, tex_box_id, str))
447     else
448         run_tex_code(format("\\luamplibtagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
449                             tex_box_id, global, tex_box_id, str))
450     end
451     local box = texgetbox(tex_box_id)
452     local wd = box.width / factor
453     local ht = box.height / factor
454     local dp = box.depth / factor
455     return text_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
456 end
457 return ""
458 end
459 end
460

```

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.

```

461 if is_defined'color_select:n' then
462     run_tex_code{
463         "\\newcatcodetable\\luamplibcctabexplat",
464         "\\begingroup",
465         "\\catcode`@=11 ",
466         "\\catcode`_=11 ",
467         "\\catcode`:=11 ",
468         "\\savecatcodetable\\luamplibcctabexplat",
469         "\\endgroup",
470     }
471 end
472 local ccexplat = luatexbase.registernumber"luamplibcctabexplat"
473
474 local process_color, process_mplibcolor

```

A common function for color functions

```

475 local function is_xcolor (str)
476     for _,v in ipairs(str:explode"!") do
477         if not v:find("^%s*%d+%s*$") and is_defined("\\color@".v) then -- priority to xcolor

```

```

478     return true
479   end
480 end
481 return false
482 end
483 local function colorsplit (res)
484   local t, tt = { }, res:explode()
485   local be = tt[1]:find"^%d" and 1 or 2
486   for i=be, #tt do
487     if not tonumber(tt[i]) then break end
488     t[#t+1] = tt[i]
489   end
490   return t
491 end
492 do
493   local colfmt = ccexplat and "l3color" or "xcolor"
494   local mplibcolorfmt = {
495     xcolor = [[{\setbox0\hbox{{\color%s\global\mplibtmptoks\expandafter{\current@color}}}}]],
496     l3color = [[{\color_export:nN%{raw}\l_tmpa_tl\mplibtmptoks\expandafter{\l_tmpa_tl}}]]
497   }
498   function process_color (str)
499     if str then
500       if not str:find("%b{") then
501         str = format("{%s}", str)
502       end
503       local myfmt = mplibcolorfmt[colfmt]
504       if colfmt == "l3color" and is_defined"color" then
505         if str:find("%b[") or is_xcolor(str:match"{{(.+)}}") then
506           myfmt = mplibcolorfmt.xcolor
507         end
508       end
509       run_tex_code(myfmt:format(str), ccexplat or catat11)
510       local t = texgettoks"mplibtmptoks"
511       if not pdfmode then
512         if t:find"^hsb" then
513           local spec = t:gsub("^hsb ", ""):gsub(" ", ",")
514           run_tex_code{"\\convertcolorspec{hsb}{", spec, "}{rgb}\\mplibtmpa"}
515           t = format("rgb %s", get_macro"mplibtmpa":gsub(",", " "))
516         elseif not t:find"%d" then -- named color
517           if not is_defined"extractcolorspecs" then err"needs xcolor package loaded" end
518           run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
519           t = format("%s %s", get_macro"mplibtmpa", get_macro"mplibtmpb":gsub(",", " "))
520         end
521         local name, value = t:match"^(%a+) (.+)"
522         if name and value then
523           local op = name == "rgb" and "rg" or name == "cmyk" and "k" or name == "gray" and "g"
524           if not op then err"unknown color model" end
525           t = ("%s %s %s %s"):format(value, op, value, op:upper())
526         end

```

```

527     elseif is_defined"ver@colorspace.sty" and t:find"^/&" then
528         local a,b,c,d = t:match"^(.- cs) (.- CS) (.- scn?) (.- SCN?)$"
529         if a and b and c and d then
530             t = tableconcat({ a, c, b, d }, " ")
531         end
532     end
533     return format('1 withprescript "mpliboverridecolor=%s"', t)
534 end
535 return ""
536 end
537 function process_mplibcolor(str)
538     local res = process_color(str)
539     if res:find" cs " then return res end
540     res = colorsplit(res:match'"mpliboverridecolor=(.)"')
541     return format("(%s)", tableconcat(res, ","))
542 end
543 end
544
545     for \mpdim or mplibdimen
546 local function process_dimen (str)
547     if str then
548         str = str:gsub("{(.+)}", "%1")
549         run_tex_code(format([[ \mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
550         return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
551     end
552     return ""
553 end

```

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

```

554 local function process_verbatimtex_text (str)
555     if str then
556         run_tex_code(str)
557     end
558     return ""
559 end
560

```

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.

```

561 local tex_code_pre_mplib = {}
562 luamplib.figid = 1
563 luamplib.in_the_fig = false
564 local function process_verbatimtex_prefig (str)
565     if str then
566         tex_code_pre_mplib[luamplib.figid] = str
567     end
568     return ""
569 end

```

```

570 local function process_verbatimtex_infig (str)
571   if str then
572     return format('special "postmplibverbtx=%s";', str)
573   end
574   return ""
575 end
576

```

For *metafun* format. see issue #79.

```

577 mp = mp or {}
578 local mp = mp
579 mp.mf_path_reset = mp.mf_path_reset or function() end
580 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
581 mp.report = mp.report or info

```

metafun 2021-03-09 changes crashes luamplib.

```

582 catcodes = catcodes or {}
583 local catcodes = catcodes
584 catcodes.numbers = catcodes.numbers or {}
585 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
586 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
587 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
588 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
589 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
590 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
591 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
592

```

Now luamplib.runscript

```

593 do
594   local runscript_funcs = {
595     luamplibtext    = process_tex_text,
596     luamplibcolor   = process_mplibcolor,
597     luamplibdimen   = process_dimen,
598     luamplibprefig  = process_verbatimtex_prefig,
599     luamplibinfig   = process_verbatimtex_infig,
600     luamplibverbtx  = process_verbatimtex_text,
601   }

```

A function from ConT_EXt general.

```

602 local function mpprint(buffer,...)
603   for i=1,select("#",...) do
604     local value = select(i,...)
605     if value ~= nil then
606       local t = type(value)
607       if t == "number" then
608         buffer[#buffer+1] = format("%.16f",value)
609       elseif t == "string" then
610         buffer[#buffer+1] = value
611       elseif t == "table" then
612         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
613       else -- boolean or whatever

```

```

614         buffer[#buffer+1] = tostring(value)
615     end
616 end
617 end
618 end
619 function luamplib.runscript (code)
620     local id, str = code:match("(.-){(.*)}")
621     if id and str then
622         local f = runscript_funcs[id]
623         if f then
624             local t = f(str)
625             if t then return t end
626         end
627     end
628     local f = loadstring(code)
629     if type(f) == "function" then
630         local buffer = {}
631         function mp.print(...)
632             mpprint(buffer,...)
633         end
634         local res = {f()}
635         buffer = tableconcat(buffer)
636         if buffer and buffer ~= "" then
637             return buffer
638         end
639         buffer = {}
640         mpprint(buffer, tableunpack(res))
641         return tableconcat(buffer)
642     end
643     return ""
644 end
645 end
646
647     luamplib.maketext
648     luamplib.legacyverbatimimtex = true
649 do

```

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

```

649     local function protecttexcontents (str)
650         return str:gsub("\\%", "\\0Percent\0")
651             :gsub("%%.-\n", "")
652             :gsub("%%.-$", "")
653             :gsub("%zPercent%z", "\\%\%")
654             :gsub("\r.-$", "")
655             :gsub("%s+", " ")
656     end
657     function luamplib.maketext (str, what)
658         if str and str ~= "" then
659             str = protecttexcontents(str)

```

```

660   if what == 1 then
661       if not str:find("\\documentclass"..name_e) and
662           not str:find("\\begin%s*{document}") and
663           not str:find("\\documentstyle"..name_e) and
664           not str:find("\\usepackage"..name_e) then
665           if luamplib.legacyverbatim then
666               if luamplib.in_the_fig then
667                   return process_verbatim_infig(str)
668               else
669                   return process_verbatim_prefig(str)
670               end
671           else
672               return process_verbatim_text(str)
673           end
674       end
675   else
676       return process_tex_text(str, true) -- bool is for 'char13'
677   end
678 end
679 return ""
680 end
681 end
682

```

luamplib's METAPOST color operators

```

683 local function get_spc_name_obj (spot)
684   if is_defined"spc@csall" then
685       for v in get_macro"spc@csall":gmatch"{{(.-)}}" do
686           local n, r = get_macro("spc@ir@"..v):match"^(.-) (%d+ 0 R)"
687           if spot == n then
688               return v, r
689           end
690       end
691   else
692       err"only l3color or colorspace is supported for spot color"
693   end
694 end
695 do
696   local function cmyk2rgb (t)
697       return {
698           1 - math.min(1, t[1]+t[4]),
699           1 - math.min(1, t[2]+t[4]),
700           1 - math.min(1, t[3]+t[4]),
701       }
702   end
703   luamplib.gettexcolor = function (str, rgb)
704       local res = process_color(str):match'"mpliboverridecolor=(.+)"'
705       if res:find" cs " then
706           info("%s is a spot color. Forced to %s", str, rgb and "RGB" or "CMYK")
707           if not is_xcolor(str) then

```

```

708     run_tex_code({
709         "\\color_export:nnN{" ,
710         str,
711         "}" ,
712         rgb and "space-sep-rgb" or "space-sep-cmyk",
713         "\\mplib@tempa",
714     },ccexplat)
715     return get_macro"mplib@tempa":explode()
716 else
717     local name, value = res:match"^(.-) cs (.-) sc"
718     local t = get_macro("spc@ascmyk@"..get_spc_name_obj(name)):explode", " -- mixed not work
719     for i = 1, #t do
720         t[i] = t[i] * value
721     end
722     return rgb and cmyk2rgb(t) or t
723 end
724 end
725 local t = colorsplit(res)
726 if #t == 3 or not rgb then return t end
727 if #t == 4 then return cmyk2rgb(t) end
728 return { t[1], t[1], t[1] }
729 end
730 end
731 luamplib.shadecolor = function (str)
732     local res = process_color(str):match'"mpliboverridecolor=(.+)"'
733     if res:find" cs " then -- spot color shade

```

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}
\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","magenta!50")
    )
  ;
endfig;
\end{mplibcode}
\end{document}

```

another one: user-defined DeviceN colorspace. This can be automatically done when process color is used as a shading color component.

```

\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}

```

```

734   local name, value, obj
735   if not is_xcolor(str) then

```

```

736 run_tex_code({ "\\color_export:nnN{", str, "{backend}\\mplib@tempa" }, ccexplat)
737 name, value = get_macro'mplib@tempa':match'{{(.-)}}{(.-)}'
738 local t = res:explode()
739 local refname = t[1]:sub(2,-1)
740 if pdfmode then
741     obj = format("%s 0 R", ltx.pdf.object_id(refname))
742 else
743     run_tex_code({ "\\mplibtmp toks\\expanded{{{\\pdf_object_ref:n{", refname, "}}}" }, ccexplat)
744     obj = texgettoks"mplibtmp toks"
745 end
746 else -- colorspace: mixing is allowed only in preamble
747     name, value = res:match"^(.-) cs (.-) sc"
748     name, obj = get_spc_name_obj(name)
749 end
750 return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
751 end
752 return format('(%s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),","), str)
753 end
754

```

luamplib.fillandstrokecolor

```

755 do
756 local function graphictextcolor (col, filldraw)
757     if col:find"^[%d%.:]+$" then
758         col = col:explode":"
759         for i=1,#col do
760             col[i] = format("%.3f", col[i])
761         end
762         local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
763         col[#col+1] = filldraw == "fill" and op or op:upper()
764         return tableconcat(col," ")
765     end
766     col = process_color(col):match'"mpliboverridecolor=(.+)"'
767     local t = col:explode()
768     local b = filldraw == "fill" and 1 or #t/2+1
769     local e = b == 1 and #t/2 or #t
770     return tableconcat(t," ", b, e)
771 end
772 function luamplib.fillandstrokecolor (fill, stroke)
773     fill = graphictextcolor(fill, "fill")
774     stroke = graphictextcolor(stroke, "stroke")
775     return format('withprescript "mpliboverridecolor=%s %s"', fill, stroke)
776 end
777 end
778

```

Remove trailing zeros for smaller PDF

```

779 local decimals = "%. %d+"
780 local function rmzeros(str) return str:gsub("%.?0+$","") end
781

```

common function for mplibgraphicstext and mpliboutlinetext

```
782 local function getrulemetric (box, curr, bp)
783   local running = -1073741824
784   local wd,ht,dp = curr.width, curr.height, curr.depth
785   wd = wd == running and box.width or wd
786   ht = ht == running and box.height or ht
787   dp = dp == running and box.depth or dp
788   if bp then
789     return wd/factor, ht/factor, dp/factor
790   end
791   return wd, ht, dp
792 end
793
```

luamplib's mplibgraphicstext operator

```
794 do
795   if not math.round then
796     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
797   end
798   local emboldenfonts = { }
799   local function roundupwidth (f, fb)
800     local wd = math.round(f.size * fb / factor * 10)
801     if wd == 0 and fb ~= 0 then
802       wd = 1
803     end
804     emboldenfonts.width = wd
805     return wd
806   end
807   local function getemboldenwidth (curr, fakebold)
808     local width = emboldenfonts.width
809     if not width then
810       local f
811       local function getglyph(n)
812         while n do
813           if n.head then
814             getglyph(n.head)
815           elseif n.font and n.font > 0 then
816             f = n.font; break
817           end
818           n = node.getnext(n)
819         end
820       end
821       getglyph(curr)
822       width = roundupwidth(font.getcopy(f or font.current()), fakebold)
823     end
824     return width
825   end
826   local function getrulewhatsit (line, wd, ht, dp)
827     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
828
```

```

828   line = line == 0 and "" or ("%f w"):format(line)
829   local pl
830   local fmt = "q %s %f %f %f %f re B Q"
831   if pdfmode then
832     pl = node.new("whatsit","pdf_literal")
833     pl.mode = 0
834   else
835     fmt = "pdf:content " ..fmt
836     pl = node.new("whatsit","special")
837   end
838   pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals,rmzeros)
839   local ss = node.new"glue"
840   node.setglue(ss, 0, 65536, 65536, 2, 2)
841   pl.next = ss
842   return pl
843 end

```

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

```

844 local tag_update_attrs
845 if is_defined"ver@tagpdf.sty" then
846   tag_update_attrs = function (n, curr)
847     while n do
848       n.attr = curr.attr
849       if n.head then
850         tag_update_attrs(n.head, curr)
851       end
852       n = node.getnext(n)
853     end
854   end
855 else
856   tag_update_attrs = function() end
857 end
858 local function embolden (box, curr, fakebold)
859   local head = curr
860   while curr do
861     if curr.head then
862       curr.head = embolden(curr, curr.head, fakebold)
863     elseif curr.replace then
864       curr.replace = embolden(box, curr.replace, fakebold)
865     elseif curr.leader then
866       if curr.leader.head then
867         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
868       elseif curr.leader.id == node.id"rule" then
869         local glue = node.effective_glue(curr, box)
870         local line = getemboldenwidth(curr, fakebold)
871         local wd,ht,dp = getrulemetric(box, curr.leader)
872         if box.id == node.id"hlist" then
873           wd = glue
874         else

```

```

875         ht, dp = 0, glue
876     end
877     local pl = getrulewhatsit(line, wd, ht, dp)
878     local pack = box.id == node.id"hlist" and node.hpack or node.vpack
879     local list = pack(pl, glue, "exactly")
880     tag_update_attrs(list, curr)
881     head = node.insert_after(head, curr, list)
882     head, curr = node.remove(head, curr)
883 end
884 elseif curr.id == node.id"rule" and curr.subtype == 0 then
885     local line = getemboldenwidth(curr, fakebold)
886     local wd, ht, dp = getrulemetric(box, curr)
887     if box.id == node.id"vlist" then
888         ht, dp = 0, ht+dp
889     end
890     local pl = getrulewhatsit(line, wd, ht, dp)
891     local list
892     if box.id == node.id"hlist" then
893         list = node.hpack(pl, wd, "exactly")
894     else
895         list = node.vpack(pl, ht+dp, "exactly")
896     end
897     tag_update_attrs(list, curr)
898     head = node.insert_after(head, curr, list)
899     head, curr = node.remove(head, curr)
900 elseif curr.id == node.id"glyph" and curr.font > 0 then
901     local f = curr.font
902     local key = format("%s:%s", f, fakebold)
903     local i = emboldenfonts[key]
904     if not i then
905         local ft = font.getfont(f) or font.getcopy(f)
906         local width = roundupwidth(ft, fakebold)
907         if ft.format == "opentype" or ft.format == "truetype" then
908             local name = ft.name:gsub("'", "'"):gsub('$', '$')
909             local t = name:gsub("^file:", ""):gsub("^name:", ""):gsub("^kpse:", ""):gsub("^my:", "")
910             name = format('%s%sembolden=%s;', name, t:find":" and ";" or ":", fakebold)
911             _, i = fonts.constructors.readanddefine(name, ft.size)
912         elseif pdfmode then
913             local ft = table.copy(ft)
914             ft.mode, ft.width = 2, width
915             i = font.define(ft)
916         else
917             goto skip_type1
918         end
919         emboldenfonts[key] = i
920     end
921     curr.font = i
922 end
923 ::skip_type1::

```

```

924     curr = node.getnext(curr)
925   end
926   return head
927 end
928 luamplib.graphicstext = function (text, fakebold, fc, dc)
929   local fmt = process_tex_text(text):sub(1,-2)
930   local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
931   emboldenfonts.width = nil
932   local box = texgetbox(id)
933   box.head = embolden(box, box.head, fakebold)
934   local colors = luamplib.fillandstrokecolor(fc, dc)
935   return format('%s %s)', fmt, colors)
936 end
937 end
938

```

luamplib's mplibglyph operator

```

939 do
940   local function mperr (str)
941     return format("hide(errmessage %q)", str)
942   end
943   local function getangle (a,b,c)
944     local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
945     if r > 180 then
946       r = r - 360
947     elseif r < -180 then
948       r = r + 360
949     end
950     return r
951   end
952   local function turning (t)
953     local r, n = 0, #t
954     for i=1,2 do
955       tableinsert(t, t[i])
956     end
957     for i=1,n do
958       r = r + getangle(t[i], t[i+1], t[i+2])
959     end
960     return r/360
961   end
962   local function glyphimage(t, fmt)
963     local q, p, r, towarn = {}, {}, {}
964     local function closepath(dots)
965       tableinsert(p, format("%scycle", dots or "--"))
966       tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
967     end
968     for i,v in ipairs(t) do
969       local cmd = v[#v]
970       local nt = t[i+1]

```

```

971     local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
972     if cmd == "m" then
973         if final then towarn = true end
974         p = {format('(%s,%s)',v[1],v[2])}
975         r = {{x=v[1],y=v[2]}}
976     else
977         if cmd == "l" then
978             local pt = t[i-1]
979             if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
980                 else
981                     tableinsert(p, format('--(%s,%s)',v[1],v[2]))
982                     tableinsert(r, {x=v[1],y=v[2]})
983                 end
984             if final then closepath() end
985         elseif cmd == "c" then
986             tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
987             if final and r[1].x == v[5] and r[1].y == v[6] then
988                 closepath ".."
989             else
990                 tableinsert(p, format('..(%s,%s)',v[5],v[6]))
991                 tableinsert(r, {x=v[5],y=v[6]})
992                 if final then closepath() end
993             end
994         elseif cmd == "path" or cmd == "move" then
995             else
996                 return mperr"unknown operator"
997             end
998         end
999     end
1000     r = { }
1001     if fmt == "opentype" then
1002         for _,v in ipairs(q[1]) do
1003             tableinsert(r, format('addto currentpicture contour %s;',v))
1004         end
1005         for _,v in ipairs(q[2]) do
1006             tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
1007         end
1008     else
1009         for _,v in ipairs(q[2]) do
1010             tableinsert(r, format('addto currentpicture contour %s;',v))
1011         end
1012         for _,v in ipairs(q[1]) do
1013             tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
1014         end
1015     end
1016     return format('image(%s)', tableconcat(r)), towarn
1017 end
1018 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
1019 function luamplib.glyph (f, c)

```

```

1020 local filename, subfont, instance, kind, shapedata
1021 local fid = tonumber(f) or font.id(f)
1022 if fid > 0 then
1023     local fontdata = font.getfont(fid) or font.getcopy(fid)
1024     filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
1025     instance = fontdata.specification and fontdata.specification.instance
1026     or fontdata.shared and fontdata.shared.features.axis
1027     filename = filename and filename:gsub("^harfloaded:", "")
1028 else
1029     local name
1030     f = f:match"^%s*(.)%s*$"
1031     name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
1032     if not name then
1033         name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
1034     end
1035     if not name then
1036         name, subfont = f:match"(.+)%((%d+)%)$" -- Times.ttc(2)
1037     end
1038     name = name or f
1039     subfont = (subfont or 0)+1
1040     instance = instance and instance:lower()
1041     for _, ftype in ipairs{"opentype", "truetype"} do
1042         filename = kpse.find_file(name, ftype.." fonts")
1043         if filename then
1044             kind = ftype; break
1045         end
1046     end
1047 end
1048 if kind ~= "opentype" and kind ~= "truetype" then
1049     f = fid and fid > 0 and tex.fontname(fid) or f
1050     if kpse.find_file(f, "tfm") then
1051         return format("glyph %s of %q", tonumber(c) or format("%q", c), f)
1052     else
1053         filename = kpse.find_file(f, "type1 fonts")
1054         if filename then
1055             kind = "type1" -- there's bug in processing cmr family
1056         else
1057             return mperr"font not found"
1058         end
1059     end
1060 end
1061 local time = lfsattributes(filename, "modification")

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

1062 local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1063 local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
1064 local newname = format("%s/%s.lua", cachedir or outputdir(), h)

```

```

1065     local newtime = lfsattributes(newname,"modification") or 0
1066     if time == newtime then
1067         shapedata = require(newname)
1068     end
1069     if not shapedata then
1070         if fonts then
1071             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1072             shapedata = handler.readers.loadshapes(filename,subfont,instance)
1073         end
1074         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1075         table.tofile(newname, shapedata, "return")
1076         lfstouch(newname, time, time)
1077     end
1078     local gid = tonumber(c)
1079     if not gid then
1080         local uni = utf8.codepoint(c)
1081         for i,v in pairs(shapedata.glyphs) do
1082             if c == v.name or uni == v.unicode then
1083                 gid = i; break
1084             end
1085         end
1086     end
1087     if not gid then return mperr"cannot get GID (glyph id)" end
1088     local fac = 1000 / (shapedata.units or 1000)
1089     local t = shapedata.glyphs[gid]; t = t and t.segments
1090     if not t then return "image()" end
1091     for i,v in ipairs(t) do
1092         if type(v) == "table" then
1093             for ii,vv in ipairs(v) do
1094                 if type(vv) == "number" then
1095                     t[i][ii] = format("%.0f", vv * fac)
1096                 end
1097             end
1098         end
1099     end
1100     local result, towarn = glyphimage(t, shapedata.format or kind)
1101     if towarn then
1102         warn("mplibglyph %s not working properly. Use glyph instead", f)
1103     end
1104     return result
1105 end
1106 end
1107

```

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

```

1108 do
1109     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1110         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1111     local outline_horz, outline_vert

```

```

1112 function outline_vert (res, box, curr, xshift, yshift)
1113     local b2u = box.dir == "LTL"
1114     local dy = (b2u and -box.depth or box.height)/factor
1115     local ody = dy
1116     while curr do
1117         if curr.id == node.id"rule" then
1118             local wd, ht, dp = getrulemetric(box, curr, true)
1119             local hd = ht + dp
1120             if hd ~= 0 then
1121                 dy = dy + (b2u and dp or -ht)
1122                 if wd ~= 0 and curr.subtype == 0 then
1123                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1124                 end
1125                 dy = dy + (b2u and ht or -dp)
1126             end
1127         elseif curr.id == node.id"glue" then
1128             local vwidth = node.effective_glue(curr,box)/factor
1129             if curr.leader then
1130                 local curr, kind = curr.leader, curr.subtype
1131                 if curr.id == node.id"rule" then
1132                     local wd = getrulemetric(box, curr, true)
1133                     if wd ~= 0 then
1134                         local hd = vwidth
1135                         local dy = dy + (b2u and 0 or -hd)
1136                         if hd ~= 0 and curr.subtype == 0 then
1137                             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1138                         end
1139                     end
1140                 elseif curr.head then
1141                     local hd = (curr.height + curr.depth)/factor
1142                     if hd <= vwidth then
1143                         local dy, n, iy = dy, 0, 0
1144                         if kind == 100 or kind == 103 then -- todo: gleaders
1145                             local ady = abs(ody - dy)
1146                             local ndy = math.ceil(ady / hd) * hd
1147                             local diff = ndy - ady
1148                             n = math.floor((vwidth-diff) / hd)
1149                             dy = dy + (b2u and diff or -diff)
1150                         else
1151                             n = math.floor(vwidth / hd)
1152                             if kind == 101 then
1153                                 local side = vwidth % hd / 2
1154                                 dy = dy + (b2u and side or -side)
1155                             elseif kind == 102 then
1156                                 iy = vwidth % hd / (n+1)
1157                                 dy = dy + (b2u and iy or -iy)
1158                             end
1159                         end
1160                     end
1161                 end
1162                 dy = dy + (b2u and curr.depth or -curr.height)/factor

```

```

1161         hd = b2u and hd or -hd
1162         iy = b2u and iy or -iy
1163         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1164         for i=1,n do
1165             res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1166             dy = dy + hd + iy
1167         end
1168     end
1169 end
1170 end
1171 dy = dy + (b2u and vwidth or -vwidth)
1172 elseif curr.id == node.id"kern" then
1173     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1174 elseif curr.id == node.id"vlist" then
1175     dy = dy + (b2u and curr.depth or -curr.height)/factor
1176     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1177     dy = dy + (b2u and curr.height or -curr.depth)/factor
1178 elseif curr.id == node.id"hlist" then
1179     dy = dy + (b2u and curr.depth or -curr.height)/factor
1180     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1181     dy = dy + (b2u and curr.height or -curr.depth)/factor
1182 end
1183 curr = node.getnext(curr)
1184 end
1185 return res
1186 end
1187 function outline_horz (res, box, curr, xshift, yshift, discwd)
1188     local r2l = box.dir == "TRT"
1189     local dx = r2l and (discwd or box.width/factor) or 0
1190     local dirs = { { dir = r2l, dx = dx } }
1191     while curr do
1192         if curr.id == node.id"dir" then
1193             local sign, dir = curr.dir:match"(.)(...)"
1194             local level, newdir = curr.level, r2l
1195             if sign == "+" then
1196                 newdir = dir == "TRT"
1197                 if r2l ~= newdir then
1198                     local n = node.getnext(curr)
1199                     while n do
1200                         if n.id == node.id"dir" and n.level+1 == level then break end
1201                         n = node.getnext(n)
1202                     end
1203                     n = n or node.tail(curr)
1204                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1205                 end
1206                 dirs[level] = { dir = r2l, dx = dx }
1207             else
1208                 local level = level + 1
1209                 newdir = dirs[level].dir

```

```

1210         if r2l ~= newdir then
1211             dx = dirs[level].dx
1212         end
1213     end
1214     r2l = newdir
1215 elseif curr.char and curr.font and curr.font > 0 then
1216     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1217     local gid = ft.characters[curr.char].index or curr.char
1218     local scale = ft.size / factor / 1000
1219     local slant = (ft.slant or 0)/1000
1220     local extend = (ft.extend or 1000)/1000
1221     local squeeze = (ft.squeeze or 1000)/1000
1222     local expand = 1 + (curr.expansion_factor or 0)/1000000
1223     local xscale, yscale = scale * extend * expand, scale * squeeze
1224     dx = dx - (r2l and curr.width/factor*expand or 0)
1225     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1226     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1227     local vertical = ""
1228     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1229         if ft.shared.features.vertical then -- luatexko
1230             vertical = "rotated 90"
1231             local data = ft.characters[curr.char] or { }
1232             if ft.hb then
1233                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1234                 local charraise = (ft.luatexko_charraise or 0)/factor
1235                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1236             else
1237                 local cmds = data.commands or { {0,0}, {0,0} }
1238                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1239                 xpos, ypos = xpos + hoff, ypos + voff
1240             end
1241         elseif curr ~= box.head then -- luatexja
1242             vertical = "rotated 90"
1243             local en = ft.parameters.quad/factor/2
1244             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1245         end
1246     end
1247     local image
1248     if ft.format == "opentype" or ft.format == "truetype" then
1249         image = luamplib.glyph(curr.font, gid)
1250     else
1251         local name, scale = ft.name, 1
1252         local vf = font.read_vf(name, ft.size)
1253         if vf and vf.characters[gid] then
1254             local cmds = vf.characters[gid].commands or { }
1255             for _,v in ipairs(cmds) do
1256                 if v[1] == "char" then
1257                     gid = v[2]
1258                 elseif v[1] == "font" and vf.fonts[v[2]] then

```

```

1259         name = vf.fonts[v[2]].name
1260         scale = vf.fonts[v[2]].size / ft.size
1261     end
1262 end
1263 end
1264 image = format("glyph %s of %q scaled %f", gid, name, scale)
1265 end
1266 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1267                     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1268 dx = dx + (r2l and 0 or curr.width/factor*expand)
1269 elseif curr.replace then
1270     local width = node.dimensions(curr.replace)/factor
1271     dx = dx - (r2l and width or 0)
1272     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1273     dx = dx + (r2l and 0 or width)
1274 elseif curr.id == node.id"rule" then
1275     local wd, ht, dp = getrulemetric(box, curr, true)
1276     if wd ~= 0 then
1277         local hd = ht + dp
1278         dx = dx - (r2l and wd or 0)
1279         if hd ~= 0 and curr.subtype == 0 then
1280             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1281         end
1282         dx = dx + (r2l and 0 or wd)
1283     end
1284 elseif curr.id == node.id"glue" then
1285     local width = node.effective_glue(curr, box)/factor
1286     dx = dx - (r2l and width or 0)
1287     if curr.leader then
1288         local curr, kind = curr.leader, curr.subtype
1289         if curr.id == node.id"rule" then
1290             local wd, ht, dp = getrulemetric(box, curr, true)
1291             local hd = ht + dp
1292             if hd ~= 0 then
1293                 wd = width
1294                 if wd ~= 0 and curr.subtype == 0 then
1295                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1296                 end
1297             end
1298         elseif curr.head then
1299             local wd = curr.width/factor
1300             if wd <= width then
1301                 local dx = r2l and dx+width or dx
1302                 local n, ix = 0, 0
1303                 if kind == 100 or kind == 103 then -- todo: gleaders
1304                     local adx = abs(dx-dirs[1].dx)
1305                     local ndx = math.ceil(adx / wd) * wd
1306                     local diff = ndx - adx
1307                     n = math.floor((width-diff) / wd)

```

```

1308         dx = dx + (r2l and -diff-wd or diff)
1309     else
1310         n = math.floor(width / wd)
1311         if kind == 101 then
1312             local side = width % wd / 2
1313             dx = dx + (r2l and -side-wd or side)
1314         elseif kind == 102 then
1315             ix = width % wd / (n+1)
1316             dx = dx + (r2l and -ix-wd or ix)
1317         end
1318     end
1319     wd = r2l and -wd or wd
1320     ix = r2l and -ix or ix
1321     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1322     for i=1,n do
1323         res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1324         dx = dx + wd + ix
1325     end
1326 end
1327 end
1328 end
1329 dx = dx + (r2l and 0 or width)
1330 elseif curr.id == node.id"kern" then
1331     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1332 elseif curr.id == node.id"math" then
1333     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1334 elseif curr.id == node.id"vlist" then
1335     dx = dx - (r2l and curr.width/factor or 0)
1336     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1337     dx = dx + (r2l and 0 or curr.width/factor)
1338 elseif curr.id == node.id"hlist" then
1339     dx = dx - (r2l and curr.width/factor or 0)
1340     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1341     dx = dx + (r2l and 0 or curr.width/factor)
1342 end
1343 curr = node.getnext(curr)
1344 end
1345 return res
1346 end
1347 function luamplib.outlinetext (text)
1348     local fmt = process_tex_text(text)
1349     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1350     local box = texgetbox(id)
1351     local res = outline_horz({ }, box, box.head, 0, 0)
1352     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1353     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1354 end
1355 end
1356

```

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

```

1357 function luamplib.getunicodegraphemes (s)
1358   local t = { }
1359   local graphemes = require'lua-uni-graphemes'
1360   for _, _, c in graphemes.graphemes(s) do
1361     table.insert(t, c)
1362   end
1363   return t
1364 end
1365 function luamplib.unicodesubstring (s,b,e,grph)
1366   local tt, t, step = { }
1367   if grph then
1368     t = luamplib.getunicodegraphemes(s)
1369   else
1370     t = { }
1371     for _, c in utf8.codes(s) do
1372       table.insert(t, utf8.char(c))
1373     end
1374   end
1375   if b <= e then
1376     b, step = b+1, 1
1377   else
1378     e, step = e+1, -1
1379   end
1380   for i = b, e, step do
1381     table.insert(tt, t[i])
1382   end
1383   s = table.concat(tt):gsub("'", "'&ditto'")
1384   return string.format("%s", s)
1385 end
1386

```

METAPOST preambles

```

1387 luamplib.preambles = {
1388   preamble = [[
1389 boolean mplib ; mplib := true ;
1390 let dump = endinput ;
1391 let normalfontsize = fontsize;
1392 input %s ;
1393 ]],
1394   mplibcode = [[
1395 texscriptmode := 2;
1396 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1397 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1398 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1399 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1400 if known context_mlib:
1401   defaultfont := "cmtt10";
1402   let infont = normalinfont;

```

```

1403 let fontsize = normalfontsize;
1404 vardef thelabel@#(expr p,z) =
1405   if string p :
1406     thelabel@#(p infont defaultfont scaled defaultscale,z)
1407   else :
1408     p shifted (z + labeloffset*mfun_laboff@# -
1409       (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1410       (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1411   fi
1412 enddef;
1413 else:
1414 vardef texttext@# primary t = rawtexttext (t) enddef;
1415 def message expr t =
1416   if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1417 enddef;
1418 def withtransparency (expr a, t) =
1419   withprescript "tr_alternative=" & if numeric a: decimal fi a
1420   withprescript "tr_transparency=" & decimal t
1421 enddef;
1422 vardef ddecimal primary p =
1423   decimal xpart p & " " & decimal ypart p
1424 enddef;
1425 vardef boundingbox primary p =
1426   if (path p) or (picture p) :
1427     llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1428   else :
1429     origin
1430   fi -- cycle
1431 enddef;
1432 fi
1433 def resolvedcolor(expr s) =
1434   runscript("return luamplib.shadecolor('"&s&"')")
1435 enddef;
1436 def colordecimals primary c =
1437   if cmykcolor c:
1438     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1439     decimal yellowpart c & ":" & decimal blackpart c
1440   elseif rgbcolor c:
1441     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1442   elseif string c:
1443     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1444   else:
1445     decimal c
1446   fi
1447 enddef;
1448 def externalfigure primary filename =
1449   draw rawtexttext("\includegraphics{"& filename &}")
1450 enddef;
1451 def TEX = texttext enddef;

```

```

1452 def mplibtexcolor primary c =
1453   runscript("return luamplib.gettexcolor('& c &''')")
1454 enddef;
1455 def mplibrbgtexcolor primary c =
1456   runscript("return luamplib.gettexcolor('& c &''', 'rgb')")
1457 enddef;
1458 def mplibgraphictext primary t =
1459   begingroup;
1460   mplibgraphictext_ (t)
1461 enddef;
1462 def mplibgraphictext_ (expr t) text rest =
1463   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1464   fb, fc, dc, graphictextpic, alsoordoublepath;
1465   picture graphictextpic; graphictextpic := nullpicture;
1466   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1467   let scale = scaled;
1468   def fakebold primary c = hide(fb:=c;) enddef;
1469   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1470   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1471   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1472   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1473   addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1474   def fakebold primary c = enddef;
1475   let fillcolor = fakebold; let drawcolor = fakebold;
1476   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1477   image(draw runscript("return luamplib.graphictext([==['&t&']==], "
1478     & decimal fb &'', '& fc &''', '& dc &''')") rest;)
1479   endgroup;
1480 enddef;
1481 def mplibglyph expr c of f =
1482   runscript (
1483     "return luamplib.glyph('"
1484     & if numeric f: decimal fi f
1485     & " ', '"
1486     & if numeric c: decimal fi c
1487     & " ')"
1488   )
1489 enddef;
1490 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1491 def mplibdrawglyph expr g =
1492   luamplib_tmp_num_ := 0;
1493   for item within g:
1494     fill pathpart item
1495     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1496   endfor
1497 enddef;
1498 let mplibfillglyph = mplibdrawglyph;
1499 def mplibstrokeglyph expr g =
1500   luamplib_tmp_num_ := 0;

```

```

1501   for item within g:
1502     draw pathpart item
1503     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1504   endfor
1505 enddef;
1506 def mplibfillandstrokeglyph expr g =
1507   luamplib_tmp_num_ := 0;
1508   for item within g:
1509     draw pathpart item withpostscript
1510     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1511   endfor
1512 enddef;
1513 def withmplibcolors (expr f, s) =
1514   runscript("return luamplib.fillandstrokecolor('" &
1515     if not string f: colordecimals fi f & "'','" &
1516     if not string s: colordecimals fi s & "'')")
1517 enddef;
1518 def withmplibopacities (expr a, f, s) =
1519   withprescript "tr_alternative=" & if numeric a: decimal fi a
1520   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1521 enddef;
1522 def mplib_do_outline_text_set_b (text f) (text d) text r =
1523   def mplib_do_outline_options_f = f enddef;
1524   def mplib_do_outline_options_d = d enddef;
1525   def mplib_do_outline_options_r = r enddef;
1526 enddef;
1527 def mplib_do_outline_text_set_f (text f) text r =
1528   def mplib_do_outline_options_f = f enddef;
1529   def mplib_do_outline_options_r = r enddef;
1530 enddef;
1531 def mplib_do_outline_text_set_u (text f) text r =
1532   def mplib_do_outline_options_f = f enddef;
1533 enddef;
1534 def mplib_do_outline_text_set_d (text d) text r =
1535   def mplib_do_outline_options_d = d enddef;
1536   def mplib_do_outline_options_r = r enddef;
1537 enddef;
1538 def mplib_do_outline_text_set_r (text d) (text f) text r =
1539   def mplib_do_outline_options_d = d enddef;
1540   def mplib_do_outline_options_f = f enddef;
1541   def mplib_do_outline_options_r = r enddef;
1542 enddef;
1543 def mplib_do_outline_text_set_n text r =
1544   def mplib_do_outline_options_r = r enddef;
1545 enddef;
1546 def mplib_do_outline_text_set_p = enddef;
1547 def mplib_fill_outline_text =
1548   for n=1 upto mpliboutlinenum:
1549     i:=0;

```

```

1550   for item within mpliboutlinepic[n]:
1551       i:=i+1;
1552       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1553       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1554   endfor
1555 endfor
1556 enddef;
1557 def mplib_draw_outline_text =
1558   for n=1 upto mpliboutlinenum:
1559       for item within mpliboutlinepic[n]:
1560           draw pathpart item mplib_do_outline_options_d;
1561       endfor
1562   endfor
1563 enddef;
1564 def mplib_filldraw_outline_text =
1565   for n=1 upto mpliboutlinenum:
1566       i:=0;
1567       for item within mpliboutlinepic[n]:
1568           i:=i+1;
1569           if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1570               fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1571           else:
1572               draw pathpart item mplib_do_outline_options_f withpostscript "both";
1573           fi
1574       endfor
1575   endfor
1576 enddef;
1577 vardef mpliboutlinetext@# (expr t) text rest =
1578   save kind; string kind; kind := str @#;
1579   save i; numeric i;
1580   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1581   def mplib_do_outline_options_d = enddef;
1582   def mplib_do_outline_options_f = enddef;
1583   def mplib_do_outline_options_r = enddef;
1584   runscript("return luamplib.outlinetext[===["&t&"]===]");
1585   image ( addto currentpicture also image (
1586       if kind = "f":
1587           mplib_do_outline_text_set_f rest;
1588           mplib_fill_outline_text;
1589       elseif kind = "d":
1590           mplib_do_outline_text_set_d rest;
1591           mplib_draw_outline_text;
1592       elseif kind = "b":
1593           mplib_do_outline_text_set_b rest;
1594           mplib_fill_outline_text;
1595           mplib_draw_outline_text;
1596       elseif kind = "u":
1597           mplib_do_outline_text_set_u rest;
1598           mplib_filldraw_outline_text;

```

```

1599     elseif kind = "r":
1600         mplib_do_outline_text_set_r rest;
1601         mplib_draw_outline_text;
1602         mplib_fill_outline_text;
1603     elseif kind = "p":
1604         mplib_do_outline_text_set_p;
1605         mplib_draw_outline_text;
1606     else:
1607         mplib_do_outline_text_set_n rest;
1608         mplib_fill_outline_text;
1609     fi;
1610 ) mplib_do_outline_options_r; )
1611 enddef ;
1612 def withmppattern primary p =
1613     withprescript "mplibpattern=" & if numeric p: decimal fi p
1614 enddef;
1615 def withpatternstroke expr a =
1616     withprescript "mplibpatternstroke=" & a
1617 enddef;
1618 primarydef t withpattern p =
1619     image(
1620         if cycle t:
1621             fill
1622         else:
1623             draw
1624         fi
1625         t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1626 enddef;
1627 vardef mplibtransformmatrix (text e) =
1628     save t; transform t;
1629     t = identity e;
1630     runscript("luamplib.transformmatrix = {"
1631     & decimal xpart t & ","
1632     & decimal ypart t & ","
1633     & decimal xpart t & ","
1634     & decimal ypart t & ","
1635     & decimal xpart t & ","
1636     & decimal ypart t & ","
1637     & "}");
1638 enddef;
1639 primarydef p withmaskinggroup s =
1640     if picture p:
1641         image(
1642             draw p;
1643             draw center p withprescript "mplibfadestate=stop";
1644         )
1645     else:
1646         p withprescript "mplibfadestate=stop"
1647     fi

```

```

1648 withprescript "mplibfadetype=masking"
1649 withprescript "mplibmaskname=" & s
1650 enddef;
1651 def withmaskingbgcolor expr c =
1652   withprescript "mplibmaskingbgcolor=" & colordecimals c
1653 enddef;
1654 primarydef p withfademethod s =
1655   if picture p:
1656     image(
1657       draw p;
1658       draw center p withprescript "mplibfadestate=stop";
1659     )
1660   else:
1661     p withprescript "mplibfadestate=stop"
1662   fi
1663   withprescript "mplibfadetype=" & s
1664   hide(mplib_shade_step := 1;)
1665   withprescript "sh_color_a=1"
1666   withprescript "sh_color_b=0"
1667   withprescript "mplibfadebbox=" &
1668     decimal (xpart llcorner p -1/4) & ":" &
1669     decimal (ypart llcorner p -1/4) & ":" &
1670     decimal (xpart urcorner p +1/4) & ":" &
1671     decimal (ypart urcorner p +1/4)
1672 enddef;
1673 def withfadevector (expr a,b) =
1674   withprescript "mplibfadevector=" &
1675     decimal xpart a & ":" &
1676     decimal ypart a & ":" &
1677     decimal xpart b & ":" &
1678     decimal ypart b
1679 enddef;
1680 let withfadecenter = withfadevector;
1681 def withfaderadius (expr a,b) =
1682   withprescript "mplibfaderadius=" &
1683     decimal a & ":" &
1684     decimal b
1685 enddef;
1686 def withfadebbox (expr a,b) =
1687   withprescript "mplibfadebbox=" &
1688     decimal xpart a & ":" &
1689     decimal ypart a & ":" &
1690     decimal xpart b & ":" &
1691     decimal ypart b
1692 enddef;
1693 primarydef p asgroup s =
1694   image(
1695     draw center p
1696     withprescript "mplibgroupbbox=" &

```

```

1697     decimal (xpart llcorner p -1/4) & ":" &
1698     decimal (ypart llcorner p -1/4) & ":" &
1699     decimal (xpart urcorner p +1/4) & ":" &
1700     decimal (ypart urcorner p +1/4)
1701     withprescript "gr_state=start"
1702     withprescript "gr_type=" & s;
1703     draw p withprescript "sh_in_xobj=yes";
1704     draw center p withprescript "gr_state=stop";
1705 )
1706 enddef;
1707 def withgroupbbox (expr a,b) =
1708   withprescript "mplibgroupbbox=" &
1709   decimal xpart a & ":" &
1710   decimal ypart a & ":" &
1711   decimal xpart b & ":" &
1712   decimal ypart b
1713 enddef;
1714 def withgroupname expr s =
1715   withprescript "mplibgroupname=" & s
1716 enddef;
1717 def usemplibgroup primary s =
1718   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s & "\endcsname}")
1719   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1720 enddef;
1721 path    mplib_shade_path ;
1722 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1723 numeric mplib_shade_fx, mplib_shade_fy ;
1724 numeric mplib_shade_lx, mplib_shade_ly ;
1725 numeric mplib_shade_nx, mplib_shade_ny ;
1726 numeric mplib_shade_dx, mplib_shade_dy ;
1727 numeric mplib_shade_tx, mplib_shade_ty ;
1728 primarydef p withshadingmethod m =
1729   p
1730   if picture p :
1731     withprescript "sh_operand_type=picture"
1732     if textual p or (length p > 1):
1733       withprescript "sh_transform=no"
1734       mplib_with_shade_method (boundingbox p, m)
1735     else:
1736       withprescript "sh_transform=yes"
1737       mplib_with_shade_method (pathpart p, m)
1738     fi
1739   else :
1740     withprescript "sh_transform=yes"
1741     mplib_with_shade_method (p, m)
1742   fi
1743 enddef;
1744 def mplib_with_shade_method (expr p, m) =
1745   hide(mplib_with_shade_method_analyze(p))

```

```

1746 withprescript "sh_domain=0 1"
1747 withprescript "sh_color=into"
1748 withprescript "sh_color_a=" & colordecimals white
1749 withprescript "sh_color_b=" & colordecimals black
1750 withprescript "sh_first=" & ddecimal point 0 of p
1751 withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1752 withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1753 if m = "linear" :
1754   withprescript "sh_type=linear"
1755   withprescript "sh_factor=1"
1756   withprescript "sh_center_a=" & ddecimal llcorner p
1757   withprescript "sh_center_b=" & ddecimal urcorner p
1758 elseif m = "coons":
1759   withprescript "sh_type=coons"
1760   withprescript "sh_transform=no"
1761 elseif m = "triangle":
1762   withprescript "sh_type=triangle"
1763   withprescript "sh_transform=no"
1764 elseif m = "lattice":
1765   withprescript "sh_type=lattice"
1766   withprescript "sh_transform=no"
1767 elseif m = "tensor":
1768   withprescript "sh_type=tensor"
1769   withprescript "sh_transform=no"
1770   withprescript "sh_tensor_path=" &
1771     ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1772     ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1773     ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1774     ddecimal 1/4[point 3 of p, point 1 of p]
1775 else :
1776   withprescript "sh_type=circular"
1777   withprescript "sh_factor=1.2"
1778   withprescript "sh_center_a=" & ddecimal center p
1779   withprescript "sh_center_b=" & ddecimal center p
1780   withprescript "sh_radius_a=" & decimal 0
1781   withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1782 fi
1783 enddef;
1784 def withlatticeverticesperrow expr n =
1785   withprescript "sh_lattice_perrow=" & decimal n
1786 enddef;
1787 def withlatticeverticesdata (text t) =
1788   hide(luamplib_tmp_num_ := 0;)
1789   withprescript "sh_lattice_data=" &
1790   for item = t:
1791     if odd(incr luamplib_tmp_num_):
1792       if pair item: ddecimal fi item & " " &
1793       fi
1794   endfor ""

```

```

1795 hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1796 for item = t:
1797   if not odd(incr luamplib_tmp_num_):
1798     withshadingstep( withshadingcolors(item,item) )
1799   fi
1800 endfor
1801 enddef;
1802 def withtrianglepatchinit (expr p, a, b, c) =
1803   if string p:
1804     withtrianglepatchnext(0, p , a)
1805     withtrianglepatchnext(0, "", b)
1806     withtrianglepatchnext(0, "", c)
1807   else:
1808     withtrianglepatchnext(0, point 0 of p, a)
1809     withtrianglepatchnext(0, point 1 of p, b)
1810     withtrianglepatchnext(0, point 2 of p, c)
1811   fi
1812 enddef;
1813 def withtrianglepatchnext (expr f, p, a) =
1814   withshadingstep(
1815     withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1816     withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1817       if string p: p else: ddecimal p fi
1818     withshadingcolors (a, a)
1819   )
1820 enddef;
1821 def withcoonspatchinit (expr p, a, b, c, d) =
1822   withshadingstep(
1823     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1824     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1825       if string p: p
1826       else:
1827         ddecimal point      0 of p & " " &
1828         ddecimal postcontrol 0 of p & " " &
1829         ddecimal precontrol  1 of p & " " &
1830         ddecimal point      1 of p & " " &
1831         ddecimal postcontrol 1 of p & " " &
1832         ddecimal precontrol  2 of p & " " &
1833         ddecimal point      2 of p & " " &
1834         ddecimal postcontrol 2 of p & " " &
1835         ddecimal precontrol  3 of p & " " &
1836         ddecimal point      3 of p & " " &
1837         ddecimal postcontrol 3 of p & " " &
1838         ddecimal precontrol  0 of p
1839       fi
1840     withshadingcolors (a, b)
1841   )
1842   withshadingstep ( withshadingcolors (c, d) )
1843 enddef;

```

```

1844 def withtensorpatchinit (expr p, q, a, b, c, d) =
1845   withshadingstep(
1846     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1847     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1848     if string p: p & " " & q
1849     else:
1850       ddecimal point      0 of p & " " &
1851       ddecimal postcontrol 0 of p & " " &
1852       ddecimal precontrol  1 of p & " " &
1853       ddecimal point      1 of p & " " &
1854       ddecimal postcontrol 1 of p & " " &
1855       ddecimal precontrol  2 of p & " " &
1856       ddecimal point      2 of p & " " &
1857       ddecimal postcontrol 2 of p & " " &
1858       ddecimal precontrol  3 of p & " " &
1859       ddecimal point      3 of p & " " &
1860       ddecimal postcontrol 3 of p & " " &
1861       ddecimal precontrol  0 of q & " " &
1862       ddecimal point      0 of q & " " &
1863       ddecimal point      1 of q & " " &
1864       ddecimal point      2 of q & " " &
1865       ddecimal point      3 of q
1866     fi
1867     withshadingcolors (a, b)
1868   )
1869   withshadingstep ( withshadingcolors (c, d) )
1870 enddef;
1871 def withcoonspatchnext (expr f, p, a, b) =
1872   withshadingstep(
1873     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1874     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1875     if string p: p
1876     else:
1877       ddecimal postcontrol 1 of p & " " &
1878       ddecimal precontrol  2 of p & " " &
1879       ddecimal point      2 of p & " " &
1880       ddecimal postcontrol 2 of p & " " &
1881       ddecimal precontrol  3 of p & " " &
1882       ddecimal point      3 of p & " " &
1883       ddecimal postcontrol 3 of p & " " &
1884       ddecimal precontrol  0 of p
1885     fi
1886     withshadingcolors (a, b)
1887   )
1888 enddef;
1889 def withtensorpatchnext (expr f, p, q, a, b) =
1890   withshadingstep(
1891     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1892     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &

```

```

1893     if string p: p & " " & q
1894     else:
1895         ddecimal postcontrol 1 of p & " " &
1896         ddecimal precontrol 2 of p & " " &
1897         ddecimal point 2 of p & " " &
1898         ddecimal postcontrol 2 of p & " " &
1899         ddecimal precontrol 3 of p & " " &
1900         ddecimal point 3 of p & " " &
1901         ddecimal postcontrol 3 of p & " " &
1902         ddecimal precontrol 0 of p & " " &
1903         ddecimal point 0 of q & " " &
1904         ddecimal point 1 of q & " " &
1905         ddecimal point 2 of q & " " &
1906         ddecimal point 3 of q
1907     fi
1908     withshadingcolors (a, b)
1909 )
1910 enddef;
1911 def mplib_with_shade_method_analyze(expr p) =
1912     mplib_shade_path := p ;
1913     mplib_shade_step := 1 ;
1914     mplib_shade_fx := xpart point 0 of p ;
1915     mplib_shade_fy := ypart point 0 of p ;
1916     mplib_shade_lx := mplib_shade_fx ;
1917     mplib_shade_ly := mplib_shade_fy ;
1918     mplib_shade_nx := 0 ;
1919     mplib_shade_ny := 0 ;
1920     mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1921     mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1922     for i=1 upto length(p) :
1923         mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1924         mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1925         if mplib_shade_tx > mplib_shade_dx :
1926             mplib_shade_nx := i + 1 ;
1927             mplib_shade_lx := xpart point i of p ;
1928             mplib_shade_dx := mplib_shade_tx ;
1929         fi ;
1930         if mplib_shade_ty > mplib_shade_dy :
1931             mplib_shade_ny := i + 1 ;
1932             mplib_shade_ly := ypart point i of p ;
1933             mplib_shade_dy := mplib_shade_ty ;
1934         fi ;
1935     endfor ;
1936 enddef;
1937 vardef mplib_max_radius(expr p) =
1938     max (
1939         (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1940         (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1941         (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),

```

```

1942    (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1943  )
1944 enddef;
1945 def withshadingstep (text t) =
1946   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1947   withprescript "sh_step=" & decimal mplib_shade_step
1948   t
1949 enddef;
1950 let withfadestep = withshadingstep;
1951 def withshadingradius expr a =
1952   withprescript "sh_radius_a=" & decimal (xpart a)
1953   withprescript "sh_radius_b=" & decimal (ypart a)
1954 enddef;
1955 def withshadingorigin expr a =
1956   withprescript "sh_center_a=" & ddecimal a
1957   withprescript "sh_center_b=" & ddecimal a
1958 enddef;
1959 def withshadingvector expr a =
1960   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1961   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1962 enddef;
1963 def withshadingdirection expr a =
1964   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1965   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1966 enddef;
1967 def withshadingtransform expr a =
1968   withprescript "sh_transform=" & a
1969 enddef;
1970 def withshadingcenter expr a =
1971   withprescript "sh_center_a=" & ddecimal (
1972     center mplib_shade_path shifted (
1973       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1974       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1975     )
1976   )
1977 enddef;
1978 def withshadingcenters (expr a, b) =
1979   withprescript "sh_center_a=" & ddecimal a
1980   withprescript "sh_center_b=" & ddecimal b
1981   withshadingtransform "no"
1982   withshadingfactor 1
1983 enddef;
1984 let withshadingpoints = withshadingcenters;
1985 def withshadingextend (expr a, b) =
1986   withprescript "sh_extend=" &
1987     if a: "true" else: "false" fi & " " &
1988     if b: "true" else: "false" fi
1989 enddef;
1990 let withfadeextend = withshadingextend;

```

```

1991 def withshadingdomain expr d =
1992   withprescript "sh_domain=" & ddecimal d
1993 enddef;
1994 def withshadingfactor expr f =
1995   withprescript "sh_factor=" & decimal f
1996 enddef;
1997 def withshadingfraction expr a =
1998   if mplib_shade_step > 0 :
1999     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
2000   fi
2001 enddef;
2002 let withfadefraction = withshadingfraction;
2003 def withshadingcolors (expr a, b) =
2004   if mplib_shade_step > 0 :
2005     withprescript "sh_color=into"
2006     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
2007     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
2008   else :
2009     withprescript "sh_color=into"
2010     withprescript "sh_color_a=" & colordecimals a
2011     withprescript "sh_color_b=" & colordecimals b
2012   fi
2013 enddef;
2014 let withfadeopacity = withshadingcolors;
2015 def withshadingstroke expr a =
2016   withprescript "sh_stroking=" & a
2017 enddef;
2018 def withshadingmatrix expr s =
2019   withprescript "sh_matrix=" & s
2020 enddef;
2021 let withfadematrix = withshadingmatrix;
2022 def mpliblength primary t =
2023   runscript("return utf8.len[==[" & t & "]==]")
2024 enddef;
2025 def mplibsubstring expr p of t =
2026   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
2027     & decimal xpart p & ",",
2028     & decimal ypart p & ")")
2029 enddef;
2030 def mplibbuclength primary t =
2031   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
2032 enddef;
2033 def mplibbuclsubstring expr p of t =
2034   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
2035     & decimal xpart p & ",",
2036     & decimal ypart p & ",true)")
2037 enddef;
2038 ]],
2039 legacyverbatimimtex = [[

```

```

2040 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
2041 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
2042 let VerbatimTeX = specialVerbatimTeX;
2043 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
2044 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
2045 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
2046 "runscript(" &ditto&
2047 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
2048 "luamplib.in_the_fig=false" &ditto& ");";
2049 ]],
2050 texttextlabel = [[
2051 let luampliboriginalinfont = infont;
2052 primarydef s infont f =
2053   if (s < char 32)
2054     or (s = char 35) % #
2055     or (s = char 36) % $
2056     or (s = char 37) % %
2057     or (s = char 38) % &
2058     or (s = char 92) % \
2059     or (s = char 94) % ^
2060     or (s = char 95) % _
2061     or (s = char 123) % {
2062     or (s = char 125) % }
2063     or (s = char 126) % ~
2064     or (s = char 127) :
2065     s luampliboriginalinfont f
2066   else :
2067     rawtexttext(s)
2068   fi
2069 enddef;
2070 def fontsize expr f =
2071   begingroup
2072     save size; numeric size;
2073     size := mplibdimen("1em");
2074     if size = 0: 10pt else: size fi
2075   endgroup
2076 enddef;
2077 ]],
2078 }
2079
process_mplibcode
When \mplibverbatim is enabled, do not expand mplibcode data.
2080 luamplib.verbatiminput = false
2081 luamplib.everymplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2082 luamplib.everyendmplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2083 function luamplib.process_mplibcode (data, instanceName)
2084   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2085 if luamplib.legacyverbatim then
2086   luamplib.figid, tex_code_pre_mplib = 1, {}
2087 end
2088 local everymplib = luamplib.everymplib[instancename]
2089 local everyendmplib = luamplib.everyendmplib[instancename]
2090 data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
2091 :gsub("\r", "\n")

```

These lines are needed for mplibverbatim mode.

```

2092 if luamplib.verbatiminput then
2093   data = data:gsub("\mpcolor%s+{.-%b{}}", "mplibcolor(\"%1\")")
2094   :gsub("\mpdim%s+{b{}}", "mplibdimen(\"%1\")")
2095   :gsub("\mpdim%s+{\\%a+}", "mplibdimen(\"%1\")")
2096   data = scan_mpcode(data, replace_texblock)
2097 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`, `verbatim` ... `etex`, and string expressions.

```

2098   local t = { } -- to store btex, verbatim, string
2099   local function store (str)
2100     t[#t+1] = str
2101     return #t
2102   end
2103   data = scan_mpcode(data, {
2104     btex = function(str)
2105       return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", store(str)) -- space
2106     end,
2107     verbatim = function(str)
2108       return format("verbatim \\unexpanded{!l!u!a!%s!m!p!l!} etex;", store(str)) -- semicolon
2109     end,
2110     str = function(str)
2111       return format("'\\unexpanded{!l!u!a!%s!m!p!l!}'", store(str))
2112     end,
2113     comment = function() return "" end,
2114     escapepercent = true,
2115   })
2116   run_tex_code(format("\mplibtmp\toks\expandafter{\expanded{%s}}", data))
2117   data = texgettoks"mplibtmp\toks"

```

Next line to address issue #55

```

2118   :gsub("##", "#")
2119   :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2120 end
2121 process(data, instancename)
2122 end
2123

```

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.

```

2124 local figcontents = { post = { } }
2125 local function put2output(a,...)
2126   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2127 end
2128 local function pdf_startfigure(n,llx,lly,urx,ury)
2129   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2130 end
2131 local function pdf_stopfigure()
2132   put2output("\mplibstoptoPDF")
2133 end

```

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

```

2134 local function pdf_literalcode (...)
2135   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2136 end
2137 local start_pdf_code = pdfmode
2138   and function() pdf_literalcode"q" end
2139   or function() put2output"\special{pdf:bcontent}" end
2140 local stop_pdf_code = pdfmode
2141   and function() pdf_literalcode"Q" end
2142   or function() put2output"\special{pdf:econtent}" end
2143

```

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

```

2144 local function put_tex_boxes (object,prescript)
2145   local box = prescript.mplibtexboxid:explode":"
2146   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2147   if n and tw and th then
2148     local op = object.path
2149     local first, second, fourth = op[1], op[2], op[4]
2150     local tx, ty = first.x_coord, first.y_coord
2151     local sx, rx, ry, sy = 1, 0, 0, 1
2152     if tw ~= 0 then
2153       sx = (second.x_coord - tx)/tw
2154       rx = (second.y_coord - ty)/tw
2155       if sx == 0 then sx = 0.00001 end
2156     end
2157     if th ~= 0 then
2158       sy = (fourth.y_coord - ty)/th
2159       ry = (fourth.x_coord - tx)/th
2160       if sy == 0 then sy = 0.00001 end
2161     end
2162

```

Attempt to address #189, the displacement issue of pdf link boxes.

```

2163   local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2164   put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2165 end
2166 end
2167

```

Colors

```
2168 local function do_preobj_CR(object,prescript)
2169   if object.postscript == "collect" then return end
2170   local override = prescript and prescript.mpliboverridecolor
2171   if override then
2172     pdf_literalcode(override)
2173     if not pdfmode and override:find" [cC][sS] " then
2174       local name1 = override:match("^/(.-) cs ")
2175       local name2 = override:match(" /(.-) CS ")
2176       if name1 then
2177         texsprint(ccexplat, {
2178           "\\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2179           name1, "}{"\\pdf_object_ref:n{" , name1, "}"
2180         })
2181       end
2182       if name2 and name1 ~= name2 then
2183         texsprint(ccexplat, {
2184           "\\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2185           name2, "}{"\\pdf_object_ref:n{" , name2, "}"
2186         })
2187       end
2188     end
2189   else
2190     local cs = object.color
2191     if cs and #cs > 0 then
2192       pdf_literalcode(luamplib.colorconverter(cs))
2193     end
2194   end
2195 end
2196
```

For transparency, shading, fading, and pattern

```
2197 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2198 local pdfobjs, pdfetcs = {}, {}
2199 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2200 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2201 pdfetcs.pgfcsp = "pgf@sys@addpdfresource@colorspaces@plain"
2202 local update_pdfobjs
2203 if pdfmode then
2204   function update_pdfobjs (os, stream)
2205     local key = os
2206     if stream then key = key..stream end
2207     local on = key and pdfobjs[key]
2208     if on then
2209       return on,false
2210     end
2211     if stream then
2212       on = pdf.immediateobj("stream",stream,os)
2213     elseif os then
```

```

2214     on = pdf.immediateobj(os)
2215 else
2216     on = pdf.reserveobj()
2217 end
2218 if key then
2219     pdfobjs[key] = on
2220 end
2221 return on,true
2222 end
2223 else
2224 function update_pdfobjs (os, stream, hex)
2225     local key = os
2226     if stream then key = key..stream end
2227     local on = key and pdfobjs[key]
2228     if on then
2229         return on,false
2230     end
2231     on = pdfetcs.cnt or 1
2232     if stream then
2233         if hex then
2234             texsprint(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2235         else
2236             texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2237         end
2238     elseif os then
2239         texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2240     else
2241         texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2242     end
2243     pdfetcs.cnt = on + 1
2244     if key then
2245         pdfobjs[key] = on
2246     end
2247     return on,true
2248 end
2249 end
2250 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2251 if pdfmode then
2252 pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2253 local getpagers = pdfetcs.getpagers
2254 local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2255 local initialize_resources = function (name)
2256     local tabname = format("%s_res",name)
2257     pdfetcs[tabname] = { }
2258     if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
2259         local obj = pdf.reserveobj()
2260         setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2261         luatexbase.add_to_callback("finish_pdffile", function()
2262             pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))

```

```

2263     end,
2264     format("luamplib.%s.finish_pdffile",name))
2265 end
2266 end
2267 pdfetcs.fallback_update_resources = function (name, res)
2268     local tabname = format("%s_res",name)
2269     if not pdfetcs[tabname] then
2270         initialize_resources(name)
2271     end
2272     if luatexbase.callbacktypes.finish_pdffile then
2273         local t = pdfetcs[tabname]
2274         t[#t+1] = res
2275     else
2276         local tpr, n = getpagers() or "", 0
2277         tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2278         if n == 0 then
2279             tpr = format("%s/%s<<%s>>", tpr, name, res)
2280         end
2281         setpagers(tpr)
2282     end
2283 end
2284 else
2285     texsprint {
2286         "\\luamplibatfirstshipout{",
2287         "\\special{pdf:obj @MPlibTr<<>>}",
2288         "\\special{pdf:obj @MPlibSh<<>>}",
2289         "\\special{pdf:obj @MPlibCS<<>>}",
2290         "\\special{pdf:obj @MPlibPt<<>>}}",
2291     }
2292 pdfetcs.fallback_update_resources = function (name,res,obj)
2293     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2294     local tabname = format("%s_res",name)
2295     if not pdfetcs[tabname] then
2296         texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2297         pdfetcs[tabname] = { }
2298     end
2299     tableinsert(pdfetcs[tabname], res)
2300 end
2301 end
2302

```

Transparency

```

2303 local function add_extgs_resources (on, new)
2304     local key = format("MPlibTr%s", on)
2305     if new then
2306         local val = format(pdfetcs.resfmt, on)
2307         if pdfmanagement then
2308             texsprint {
2309                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"

```

```

2310     }
2311 else
2312     local tr = format("/%s %s", key, val)
2313     if is_defined(pdfetcs.pgfbextgs) then
2314         texsprint { "\\csname ", pdfetcs.pgfbextgs, "\\endcsname{", tr, "}" }
2315     elseif is_defined"TRP@list" then
2316         texsprint(catat11,{
2317             [[\if@files\immediate\write\@auxout{]],
2318             [[\string\g@addto@macro\string\TRP@list{]],
2319             tr,
2320             [[}}\fi]],
2321         })
2322         if not get_macro"TRP@list":find(tr) then
2323             texsprint(catat11,[[\global\TRP@runtrue]])
2324         end
2325     else
2326         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2327     end
2328 end
2329 end
2330 return key
2331 end
2332
2333 local do_preobj_TR
2334 do
2335     local transparency_modes = {
2336         [0] = "Normal",
2337         "Normal",      "Multiply",      "Screen",      "Overlay",
2338         "SoftLight",    "HardLight",    "ColorDodge",  "ColorBurn",
2339         "Darken",       "Lighten",     "Difference",  "Exclusion",
2340         "Hue",           "Saturation",  "Color",       "Luminosity",
2341         "Compatible",
2342         normal      = "Normal",    multiply = "Multiply",    screen   = "Screen",
2343         overlay     = "Overlay",    softlight = "SoftLight",  hardlight = "HardLight",
2344         colordodge = "ColorDodge",  colorburn = "ColorBurn",  darken   = "Darken",
2345         lighten     = "Lighten",    difference = "Difference", exclusion = "Exclusion",
2346         hue          = "Hue",        saturation = "Saturation", color     = "Color",
2347         luminosity = "Luminosity", compatible = "Compatible",
2348     }
2349     function do_preobj_TR(object,prescript)
2350         if object.postscript == "collect" then return end
2351         local opaq = prescript and prescript.tr_transparency
2352         if not opaq then return end
2353
2354         local key, on, os, new
2355         local mode = prescript.tr_alternative or 1
2356         mode = transparency_modes[tonumber(mode) or mode:lower()]
2357         if not mode then
2358             mode = prescript.tr_alternative

```

```

2359     warn("unsupported blend mode: '%s'", mode)
2360 end
2361 opaq = opaq:explode":""
2362 for i,v in ipairs(opaq) do
2363     opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2364 end
2365 for i,v in ipairs[ {mode,opaq[1],opaq[2] or opaq[1]},{ "Normal",1,1} ] do
2366     os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2367     on, new = update_pdfobjs(os)
2368     key = add_extgs_resources(on,new)
2369     if i == 1 then
2370         pdf_literalcode("/%s gs",key)
2371     else
2372         return format("/%s gs",key)
2373     end
2374 end
2375 end
2376 end
2377

```

Shading with *metafun* format.

```

2378 local function add_shading_resources (on, new)
2379     if new then
2380         local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2381         if pdfmanagement then
2382             texsprint {
2383                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2384             }
2385         else
2386             local res = format("/%s %s", key, val)
2387             pdfetcs.fallback_update_resources("Shading",res,"@MPlibSh")
2388         end
2389     end
2390 end
2391 local function sh_pdfpageresources(shtype,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2392     for _,v in ipairs{ca,cb} do
2393         for i,vv in ipairs(v) do
2394             for ii,vvv in ipairs(vv) do
2395                 v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2396             end
2397         end
2398     end
2399     local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2400     if steps > 1 then
2401         local list,bounds,encode = { },{ },{ }
2402         for i=1,steps do
2403             if i < steps then
2404                 bounds[i] = format("%.3f", fractions[i] or 1)
2405             end
2406         end
2407     end
2408 end

```

```

2406     encode[2*i-1] = 0
2407     encode[2*i]   = 1
2408     os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
2409         :gsub(decimals,rmzeros)
2410     list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2411 end
2412 os = tableconcat {
2413     "<</FunctionType 3",
2414     format("/Bounds[%s]",   tableconcat(bounds, ' ')),
2415     format("/Encode[%s]",   tableconcat(encode, ' ')),
2416     format("/Functions[%s]", tableconcat(list, ' ')),
2417     format("/Domain[%s]>>", domain),
2418 } :gsub(decimals,rmzeros)
2419 else
2420     os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
2421         :gsub(decimals,rmzeros)
2422 end
2423 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2424 os = tableconcat {
2425     format("<</ShadingType %i", shtype),
2426     format("/ColorSpace %s",   colorspace),
2427     format("/Function %s",     objref),
2428     format("/Coords[%s]",     coordinates),
2429     format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2430 } :gsub(decimals,rmzeros)
2431 local on, new = update_pdfobjs(os)
2432 add_shading_resources(on, new)
2433 return on
2434 end
2435
2436 local function get_mp_matrix (matrix)
2437     process(("mplibtransformmatrix(%s);"):format(matrix), "@mplibtransformmatrix")
2438     return luamplib.transformmatrix
2439 end
2440
2441 local do_preobj_SH
2442 do
2443     pdfetcs.clrspcs = setmetatable({}, { __index = function(t,names)
2444         run_tex_code({
2445             [{"\color_model_new:nnn}],
2446             format("{mplibcolorspace_%s}", names:gsub(",","_")),
2447             format("{DeviceN}{names={%s}}", names),
2448             [{"\mplibtmptoks\expanded{ {\pdf_object_ref_last:} }"]},
2449         }, ccexplat)
2450         local colorspace = texgettoks"mplibtmptoks"
2451         t[names] = colorspace
2452         return colorspace
2453     end })
2454     local function color_normalize(ca,cb)

```

```

2455     if #cb == 1 then
2456         if #ca == 4 then
2457             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2458         else -- #ca = 3
2459             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2460         end
2461     elseif #cb == 3 then -- #ca == 4 : rgb to cmyk
2462         local c, m, y = 1-cb[1], 1-cb[2], 1-cb[3]
2463         local k = math.min(c, m, y)
2464         if k == 1 then
2465             c, m, y = 0, 0, 0
2466         else
2467             c, m, y = (c-k)/(1-k), (m-k)/(1-k), (y-k)/(1-k)
2468         end
2469         cb[1], cb[2], cb[3], cb[4] = c, m, y, k
2470     end
2471 end
2472 local function write_mesh_objs (shdtype, colorspace, stream, devicen, perrow)
2473     local cc = colorspace == "/DeviceCMYK" and 4
2474         or colorspace == "/DeviceRGB" and 3
2475         or devicen or 1
2476     local dict = format(
2477         "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2478         shdtype,
2479         perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2480         colorspace,
2481         (" 0 1"):rep(cc))
2482     local on, new
2483     if pdfmode then
2484         on, new = update_pdfobjs(dict, stream)
2485     else
2486         stream = format("%02X"):rep(stream:len()), stream:byte(1,stream:len())
2487         on, new = update_pdfobjs(dict, stream, true) -- hex is true
2488     end
2489     add_shading_resources(on, new)
2490     return on
2491 end
2492 local function colors_ff (colors)
2493     local t, devicen = { }
2494     for i,v in ipairs(colors) do
2495         t[i] = { }
2496         for _,vv in ipairs(v) do
2497             local tt = tableconcat(vv," "):explode()
2498             for _,vvv in ipairs(tt) do
2499                 tableinsert(t[i], string.char(math.floor(vvv * 0xFF)))
2500             end
2501             devicen = #tt
2502         end
2503     end

```

```

2504     return t, devicen
2505 end
2506 local function coords_ffff (coords)
2507     local Xt, Yt = { }, { }
2508     for i,v in ipairs(coords) do
2509         for ii,vv in ipairs(v) do
2510             local t = ii % 2 == 1 and Xt or Yt
2511             t[#t+1] = tonumber(vv)
2512         end
2513     end
2514     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2515     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2516     local wd, ht = xmax - xmin, ymax - ymin
2517     for i,v in ipairs(coords) do
2518         for ii,vv in ipairs(v) do
2519             local min = ii % 2 == 1 and xmin or ymin
2520             local dim = ii % 2 == 1 and wd or ht
2521             coords[i][ii] = string.pack(">H", math.floor((vv - min)/dim * 0xFFFF ))
2522         end
2523     end
2524     return coords, xmin, ymin, wd, ht
2525 end
2526 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2527     local perrow = prescript.sh_lattice_perrow or 2
2528     local steps = tonumber(prescript.sh_step) or 0
2529     local coords, colors = { }, { }
2530     if steps < 4 then
2531         local path = object.path
2532         for _,i in ipairs{1,2,4,3} do
2533             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2534         end
2535         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}}, {{1,1,0}} }
2536         colorspace = "/DeviceRGB"
2537     else
2538         local t = prescript.sh_lattice_data:explode()
2539         for i = 1, #t, 2 do
2540             coords[#coords+1] = { t[i], t[i+1] }
2541         end
2542         for i = 1, steps do
2543             if not ca[i] then err"colorspace mismatch?" end
2544             colors[#colors+1] = { ca[i] }
2545         end
2546     end
2547     if #coords % perrow ~= 0 then err"vertices_per_row mismatches lattice_coords" end
2548
2549     local stream, xmin, ymin, wd, ht, devicen = { }
2550     coords, xmin, ymin, wd, ht = coords_ffff(coords)
2551     colors, devicen = colors_ff(colors)
2552     for i = 1, #coords do

```

```

2553     stream[#stream+1] = tableconcat(coords[i])
2554     stream[#stream+1] = tableconcat(colors[i])
2555 end
2556
2557 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2558 local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2559 return on, matrix
2560 end
2561 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2562     local steps = tonumber(prescript.sh_step) or 0
2563     local coords, colors, edges = { }, { }, { }
2564     if steps < 3 then
2565         local path = object.path
2566         for i = 1, 3 do
2567             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2568         end
2569         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}} }
2570         edges = { 0, 0, 0 }
2571         colorspace = "/DeviceRGB"
2572     else
2573         for i = 1, steps do
2574             local t = prescript["sh_triangle_vertex_" .. i]:explode()
2575             if #t == 2 then
2576                 coords[#coords+1] = { t[1], t[2] }
2577             elseif #t == 6 then
2578                 coords[#coords+1] = { t[1], t[2] }
2579                 coords[#coords+1] = { t[3], t[4] }
2580                 coords[#coords+1] = { t[5], t[6] }
2581             end
2582             if not ca[i] then err"colorspace mismatch?" end
2583             colors[#colors+1] = { ca[i] }
2584             edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])
2585         end
2586     end
2587
2588     local stream, xmin, ymin, wd, ht, devicen = { }
2589     coords, xmin, ymin, wd, ht = coords_ffff(coords)
2590     colors, devicen = colors_ff(colors)
2591     for i = 1, #coords do
2592         stream[#stream+1] = string.char(edges[i])
2593         stream[#stream+1] = tableconcat(coords[i])
2594         stream[#stream+1] = tableconcat(colors[i])
2595     end
2596
2597     local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2598     local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2599     return on, matrix
2600 end
2601 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)

```

```

2602 local tensor = prescript.sh_type == "tensor"
2603 local steps = tonumber(prescript.sh_step) or 0
2604 local coords, colors, edges = { }, { }, { }
2605 if steps < 2 then
2606     local path, t = object.path, { }
2607     for i = 1, 4 do
2608         t[#t+1] = path[i].x_coord
2609         t[#t+1] = path[i].y_coord
2610         t[#t+1] = path[i].right_x
2611         t[#t+1] = path[i].right_y
2612         local j = i == 4 and 1 or i+1
2613         t[#t+1] = path[j].left_x
2614         t[#t+1] = path[j].left_y
2615     end
2616     if tensor then
2617         t = table.move(prescript.sh_tensor_path:explode(), 1, 8, #t+1, t)
2618     end
2619     coords = { t }
2620     colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2621     edges = { 0 }
2622     colorspace = "/DeviceRGB"
2623 else
2624     for i = 1, steps do
2625         local edge = tonumber(prescript["sh_coons_edge_"..i])
2626         if edge then
2627             edges[#edges+1] = edge
2628             coords[#coords+1] = prescript["sh_coons_path_"..i]:explode()
2629             if edge == 0 then
2630                 if not (ca[i] and cb[i] and ca[i+1] and cb[i+1]) then err"colorspace mismatch?" end
2631                 colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2632             else
2633                 if not (ca[i] and cb[i]) then err"colorspace mismatch?" end
2634                 colors[#colors+1] = { ca[i], cb[i] }
2635             end
2636         end
2637     end
2638 end
2639
2640 local stream, xmin, ymin, wd, ht, devicen = { }
2641 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2642 colors, devicen = colors_ff(colors)
2643 for i = 1, #coords do
2644     stream[#stream+1] = string.char(edges[i])
2645     stream[#stream+1] = tableconcat(coords[i])
2646     stream[#stream+1] = tableconcat(colors[i])
2647 end
2648
2649 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2650 local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)

```

```

2651     return on, matrix
2652 end
2653 function do_preobj_SH(object, prescript)
2654     local shade_no
2655     local sh_type = prescript and prescript.sh_type
2656     if not sh_type then return end
2657
2658     local domain = prescript.sh_domain or "0 1"
2659     local centera = (prescript.sh_center_a or "0 0"):explode()
2660     local centerb = (prescript.sh_center_b or "0 0"):explode()
2661     local transform = prescript.sh_transform == "yes"
2662     local sx,sy,sr,dx,dy = 1,1,1,0,0
2663     if transform then
2664         local first = (prescript.sh_first or "0 0"):explode()
2665         local setx = (prescript.sh_set_x or "0 0"):explode()
2666         local sety = (prescript.sh_set_y or "0 0"):explode()
2667         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2668         if x ~= 0 and y ~= 0 then
2669             local path = object.path
2670             local path1x = path[1].x_coord
2671             local path1y = path[1].y_coord
2672             local path2x = path[x].x_coord
2673             local path2y = path[y].y_coord
2674             local dxa = path2x - path1x
2675             local dya = path2y - path1y
2676             local dxb = setx[2] - first[1]
2677             local dyb = sety[2] - first[2]
2678             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2679                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
2680                 sy = dya / dyb ; if sy < 0 then sy = - sy end
2681                 sr = math.sqrt(sx^2 + sy^2)
2682                 dx = path1x - sx*first[1]
2683                 dy = path1y - sy*first[2]
2684             end
2685         end
2686     end
2687     local ca, cb, colorspace, steps, fractions
2688     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:" }
2689     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:" }
2690     steps = tonumber(prescript.sh_step) or 1
2691     if steps > 1 then
2692         fractions = { prescript.sh_fraction_1 or 0 }
2693         for i=2,steps do
2694             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2695             ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
2696             cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
2697         end
2698     end
2699     if prescript.mplib_spotcolor then

```

```

2700 ca, cb = { }, { }
2701 local names, pos, objref = { }, -1, ""
2702 local script = object.prescript:explode"\13+"
2703 for i=#script,1,-1 do
2704     local name, value
2705     if script[i]:find"mplib_spotcolor" then
2706         local t = script[i]:explode"="[2]:explode":"
2707         value, objref, name = t[1], t[2], t[3]
2708     elseif script[i]:find"mplib_texcolor" then
2709         local t = script[i]:explode"="[2]:explode"!"
2710         name, value = t[1], (t[2] or 100)/100
2711     end
2712     if name and value then
2713         if not names[name] then
2714             pos = pos+1
2715             names[name] = pos
2716             names[#names+1] = name
2717         end
2718         local t = { }
2719         for j=1,names[name] do t[#t+1] = 0 end
2720         t[#t+1] = value
2721         tableinsert(#ca == #cb and ca or cb, t)
2722     end
2723 end
2724 for _,t in ipairs{ca,cb} do
2725     for _,tt in ipairs(t) do
2726         for i=1,#names-#tt do tt[#tt+1] = 0 end
2727     end
2728 end
2729 if #names == 1 then
2730     colorspace = objref
2731 else
2732     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2733 end
2734 else
2735     local model = 0
2736     for _,t in ipairs{ca,cb} do
2737         for _,tt in ipairs(t) do
2738             model = model > #tt and model or #tt
2739         end
2740     end
2741     for _,t in ipairs{ca,cb} do
2742         for _,tt in ipairs(t) do
2743             if #tt < model then
2744                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2745             end
2746         end
2747     end
2748     colorspace = model == 4 and "/DeviceCMYK"

```

```

2749         or model == 3 and "/DeviceRGB"
2750         or model == 1 and "/DeviceGray"
2751         or err"unknown color model"
2752     end
2753     local extend = prescript.sh_extend
2754     local mesh_matrix
2755     if sh_type == "linear" then
2756         local coordinates = format("%f %f %f %f",
2757             dx + sx*centera[1], dy + sy*centera[2],
2758             dx + sx*centerb[1], dy + sy*centerb[2])
2759         shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2760     elseif sh_type == "circular" then
2761         local factor = prescript.sh_factor or 1
2762         local radiusa = factor * prescript.sh_radius_a
2763         local radiusb = factor * prescript.sh_radius_b
2764         local coordinates = format("%f %f %f %f %f %f",
2765             dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2766             dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2767         shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2768     elseif sh_type == "coons" or sh_type == "tensor" then
2769         shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2770     elseif sh_type == "triangle" then
2771         shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2772     elseif sh_type == "lattice" then
2773         shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2774     else
2775         err"unknown shading type"
2776     end
2777
2778     local matrix = prescript.sh_matrix
2779     if matrix and matrix:find"%a" then
2780         local t = get_mp_matrix(matrix)
2781         matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2782     end
2783     if mesh_matrix and matrix then
2784         local a, b = mesh_matrix:explode(), matrix:explode()
2785         matrix = format("%f %f %f %f %f %f",
2786             a[1]*b[1]+a[2]*b[3], a[1]*b[2]+a[2]*b[4], a[3]*b[1]+a[4]*b[3], a[3]*b[2]+a[4]*b[4],
2787             a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)
2788     end
2789
2790     return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix
2791 end
2792 end
2793

```

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

```

2794 local function add_pattern_resources (key, val)
2795     if pdfmanagement then

```

```

2796     texsprint {
2797         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2798     }
2799     else
2800         local res = format("/%s %s", key, val)
2801         if is_defined(pdfetcs.pgfpattern) then
2802             texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2803         else
2804             pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2805         end
2806     end
2807 end
2808 if pdfmode then
2809     function luamplib.dolatelua (on, os, matrix)
2810         local h, v = pdf.getpos()
2811         local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2812         matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2813         :gsub(decimals,rmzeros)
2814         pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2815         pdf.refobj(on)
2816     end
2817 else
2818     pdfetcs.shadingpatterns = { }
2819     pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2820     function luamplib.dolatelua (on, kind, xobj)
2821         local h, v
2822         local t = pdfetcs.shadingpatterns[on] or { }
2823         local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2824             or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2825         if shift then
2826             h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2827         else
2828             h, v = pdf.getpos()
2829             h = format("%f", h/factor) :gsub(decimals,rmzeros)
2830             v = format("%f", v/factor) :gsub(decimals,rmzeros)
2831         end
2832         if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2833             warn"Rerun to get correct shading pattern"
2834         end
2835         local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2836         local init = pdfetcs.shadingpatterninit_w
2837         if init then pdfetcs.shadingpatterninit_w = nil end
2838         local f = ioopen(name, init and "w" or "a")
2839         if f then
2840             f:write(("<<%s %s %s\n"):format(on, h, v))
2841             f:close()
2842         else
2843             err"cannot write a file. check the cache dir path"
2844         end
2845     end
2846 end

```

```

2845 end
2846 end
2847 local function do_preobj_shading (object, prescript)
2848   if not prescript or not prescript.sh_operand_type then return end
2849   local on,_,matrix = do_preobj_SH(object, prescript)
2850   local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2851   matrix = matrix or "1 0 0 1 0 0"
2852   if prescript.sh_in_xobj == "yes" then
2853     on = update_pdfobjs(("<<%s/Matrix[%s]>>"):format(os, matrix))
2854     goto skip_latelua
2855   end
2856   on = update_pdfobjs()
2857   if pdfmode then
2858     put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"],[",matrix,"]])}")
2859   else
2860     local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2861               or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2862     local init = pdfetcs.shadingpatterninit_r
2863     if init then
2864       pdfetcs.shadingpatterninit_r = nil
2865       local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2866       local f = ioopen(name)
2867       if f then
2868         for line in f:lines() do
2869           local t = line:explode()
2870           pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2871         end
2872         f:close()
2873       end
2874     end
2875   end

```

This seems to be needed for proper functioning:

```

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

2875   local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2876   local mt = matrix:explode()
2877   matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2878   texsprint{ "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2879             format(" <<%s/Matrix[%s]>>", os, matrix) }
2880   put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2881             xobj and ("%s',[[%s]]"):format(xobj[1], xobj[2]))
2882 end
2883 ::skip_latelua::
2884 local key, val = format("MPLibPt%s", on), format(pdfetcs.resfmt, on)
2885 add_pattern_resources(key,val)

```

When withshadingstroke is filldraw or both, we shade the fill and the stroke; when withshadingstroke is draw or yes, we shade the stroke; all other cases shade the fill, the default behavior.

```

2886 local stroke = prescript.sh_stroking
2887 if stroke == "filldraw" or stroke == "both" then
2888   pdf_literalcode("/Pattern cs/%s scn /Pattern CS/%s SCN", key, key)
2889 elseif stroke == "draw" or stroke == "yes" then
2890   pdf_literalcode("/Pattern CS/%s SCN", key)
2891 else
2892   pdf_literalcode("/Pattern cs/%s scn", key)
2893 end

```

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

```

2894 prescript.sh_type = nil
2895 end
2896

```

Tiling Patterns

```

2897 pdfetcs.patterns = { }
2898 local function gather_resources (optres, ispattern)
2899   local t = { }
2900   if pdfmanagement then
2901     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2902       local mytoks
2903       run_tex_code ({
2904         "\\mplibtmptoks\\expanded{",
2905         "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/", v, "}",
2906         "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/", v, "}}", "}" },
2907         ccexplat)
2908       mytoks = texgettoks"mplibtmptoks"
2909       if not pdfmode then
2910         mytoks = mytoks:gsub("\\str_convert_pdfname:n%s*{(.-)}", "%1") -- why not expanded?
2911       end
2912       mytoks = mytoks and mytoks:gsub("^%s*(.)%s*$", "%1")
2913       if mytoks and mytoks ~= "" then
2914         t[#t+1] = ("%s<<%s>>"):format(v, mytoks)
2915       end
2916     end
2917   elseif is_defined(pdfetcs.pgfbextgs) then
2918     run_tex_code "\\relax" -- flush tex.sprint queue
2919     if pdfmode then
2920       for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2921                         ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2922                         Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2923         local res = (get_macro(v) or ""):gsub("^%s*(.)%s*$", "%1")
2924         if res ~= "" then
2925           t[#t+1] = ("%s<<%s>>"):format(k, res )
2926         end
2927       end
2928     else
2929       local abc = get_macro"pgfutil@abc" or ""
2930       for k,v in pairs { ExtGState = "@pgfbextgs",
2931                         ColorSpace = "@pgfcolorspaces",

```

```

2932         Pattern      = "@pgfpatterns", } do
2933     local tt = { }
2934     for vv in abc:gmatch( v .. "%s*(%b<>)" ) do
2935         tt[#tt+1] = vv:match("^<<%s*(.)%s*>>$")
2936     end
2937     if #tt > 0 then
2938         t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2939     end
2940 end
2941 end

```

We still have to deal with Shading resources.

```

2942 if luatexbase.callbacktypes.finish_pdffile then
2943     if pdfetcs.Shading_res then
2944         t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2945     end
2946 else
2947     local res = pdfetcs.getpageres()
2948     res = res and res:match"/Shading%s*%b<>"
2949     if res then
2950         t[#t+1] = res
2951     end
2952 end
2953 else
2954     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

```

2955     warn"transparent package is not fully functional without pdfmanagement code."
2956 end
2957 if luatexbase.callbacktypes.finish_pdffile then
2958     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2959         local tt = pdfetcs[v.."_res"]
2960         if tt then
2961             t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2962         end
2963     end
2964 else
2965     local res = pdfetcs.getpageres()
2966     if res then
2967         t[#t+1] = res
2968     end
2969 end
2970 end
2971 local result = tableconcat(t)
2972 if optres ~= "" then
2973     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2974         local res = optres:match("/"..v.."%.%.%.%s*%b<>")
2975         if res then
2976             if result:find("/"..v) then

```

```

2977         res = res:match("<<(.+)>>$")
2978         result = result:gsub("/"..v.."s*<<", "%1"..res, 1)
2979     else
2980         result = result .. res
2981     end
2982 end
2983 end
2984 end
2985 return result
2986 end
2987 function luamplib.registerpattern ( boxid, name, opts )
2988     local box = texgetbox(boxid)
2989     local wd = format("%.3f",box.width/factor)
2990     local hd = format("%.3f", (box.height+box.depth)/factor)
2991     info("w/h/d of pattern 's': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2992     if opts.xstep == 0 then opts.xstep = nil end
2993     if opts.ystep == 0 then opts.ystep = nil end
2994     if opts.colored == nil then
2995         opts.colored = opts.coloured
2996         if opts.colored == nil then
2997             opts.colored = true
2998         end
2999     end
3000     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
3001     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
3002     if opts.matrix and opts.matrix:find"%a" then
3003         local t = get_mp_matrix(opts.matrix)
3004         opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
3005         opts.xshift = opts.xshift or format("%f",t[5])
3006         opts.yshift = opts.yshift or format("%f",t[6])
3007     end
3008     local attr = {
3009         "/Type/Pattern",
3010         "/PatternType 1",
3011         format("/PaintType %i", opts.colored and 1 or 2),
3012         "/TilingType 2",
3013         format("/XStep %s", opts.xstep or wd),
3014         format("/YStep %s", opts.ystep or hd),
3015         format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
3016     }
3017     local optres = opts.resources or ""
3018     optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
3019     local patterns = pdfetcs.patterns
3020     if pdfmode then
3021         if opts.bbox then
3022             attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3023         end
3024         attr = tableconcat(attr) :gsub(decimals,rmzeros)
3025         local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)

```

```

3026     patterns[name] = { id = index, colored = opts.colored }
3027 else
3028     local cnt = #patterns + 1
3029     local objname = "@mplibpattern" .. cnt
3030     local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
3031     texsprint {
3032         "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
3033         "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
3034         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3035         "\\special{pdf:bcontent}",
3036         "\\special{pdf:bxobj ", objname, " ", metric, "}",
3037         "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
3038         "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
3039         "\\special{pdf:put @resources <<", optres, ">>}",
3040         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3041         "\\special{pdf:econtent}}",
3042     }
3043     patterns[cnt] = objname
3044     patterns[name] = { id = cnt, colored = opts.colored }
3045     patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
3046 end
3047 end
3048
3049 local do_preobj_PAT
3050 do
3051     local function pattern_colorspace (cs)
3052         local on, new = update_pdfobjs(format("[Pattern %s]", cs))
3053         if new then
3054             local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
3055             if pdfmanagement then
3056                 texsprint {
3057                     "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
3058                 }
3059             else
3060                 local res = format("/%s %s", key, val)
3061                 if is_defined(pdfetcs.pgfcolorspace) then
3062                     texsprint { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
3063                 else
3064                     pdfetcs.fallback_update_resources("ColorSpace",res,"@MPlibCS")
3065                 end
3066             end
3067         end
3068         return on
3069     end
3070     local function uncoloredGS(color,key,fill)
3071         local cs
3072         if color:find" cs " then
3073             local name
3074             if fill then

```

```

3075     name, color = color:match"^/(.-) cs (.-) sc"
3076   else
3077     name, color = color:match" /(-) CS (-) SC"
3078   end
3079   if pdfmode then
3080     cs = format("%s 0 R", ltx.pdf.object_id( name ))
3081     if cs == "0 0 R" then -- assumes colorspace.sty
3082       _, cs = get_spc_name_obj("/"..name)
3083     end
3084   else
3085     run_tex_code({ "\\mplibmptoks\\expanded{{{\\pdf_object_ref:n{", name, "}}}" }, ccexplat)
3086     cs = texgettoks"mplibmptoks"
3087   end
3088   else
3089     if fill then
3090       color = colorsplit(color)
3091     else
3092       color = color:match"%a+ (-) %a+":explode()
3093     end
3094     cs = #color == 4 and "/DeviceCMYK" or #color == 3 and "/DeviceRGB" or "/DeviceGray"
3095     color = tableconcat(color, " ")
3096   end
3097   return fill and format("/MPlibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3098     or format("/MPlibCS%i CS %s /%s SCN", pattern_colorspace(cs), color, key)
3099 end
3100 function do_preobj_PAT(object, prescript)
3101   local name = prescript and prescript.mplibpattern
3102   if not name then return end
3103   local patterns = pdfetcs.patterns
3104   local patt = patterns[name]
3105   local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3106   local key = format("MPlibPt%s", index)
3107   local stroke, fillgs, strkgs = prescript.mplibpatternstroke
3108   if patt.colored then
3109     fillgs = format("/Pattern cs /%s scn", key)
3110     strkgs = stroke and format("/Pattern CS /%s SCN", key)
3111   else
3112     local color = prescript.mpliboverridecolor
3113     if not color then
3114       local t = object.color
3115       color = t and #t>0 and luamplib.colorconverter(t)
3116     end
3117     if not color then return end
3118     fillgs = uncoloredGS(color, key, true)
3119     strkgs = stroke and uncoloredGS(color, key, false)
3120   end

```

When withpatternstroke is filldraw or both, we tile the fill and the stroke; when withpatternstroke is draw or yes, we tile the stroke; all other cases tile the fill, the default behavior.

```

3121   if stroke == "filldraw" or stroke == "both" then
3122     pdf_literalcode("%s %s", fillgs, strkgs)
3123   elseif stroke == "draw" or stroke == "yes" then
3124     pdf_literalcode(strkgs)
3125   else
3126     pdf_literalcode(fillgs)
3127   end
3128   if not patt.done then
3129     local val = pdfmode and format("%s 0 R",index) or patterns[index]
3130     add_pattern_resources(key,val)
3131   end
3132   patt.done = true
3133 end
3134 end
3135

```

Fading

```

3136 pdfetcs.fading = { }
3137 local function do_preobj_FADE (object, prescript)
3138   local fd_type = prescript and prescript.mplibfadetype
3139   local fd_stop = prescript and prescript.mplibfadestate
3140   if not fd_type then
3141     return fd_stop -- returns "stop" (if picture) or nil
3142   end
3143   local on, os, new
3144   if fd_type == "masking" then
3145     local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3146     on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
3147     local bc = prescript.mplibmaskingbgcolor
3148     bc = bc and bc:gsub(":", " ")
3149     bc = bc and ("BC[%s]"):format(bc):gsub(decimals,rmzeros) or ""
3150     os = format("<</SMask<</S/Luminosity/G %s%>>>>",
3151               pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3152   else
3153     local bbox = prescript.mplibfadebbox:explode":"
3154     local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
3155     if not vec then
3156       if fd_type == "linear" then
3157         vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3158       else
3159         local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3160         vec = {centerx, centery, centerx, centery} -- center for both circles
3161       end
3162     end
3163     local coords = { vec[1], vec[2], vec[3], vec[4] }
3164     if fd_type == "linear" then
3165       coords = format("%f %f %f %f", tableunpack(coords))
3166     elseif fd_type == "circular" then
3167       local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]

```

```

3168     local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":""
3169     tableinsert(coords, 3, radius[1])
3170     tableinsert(coords, radius[2])
3171     coords = format("%f %f %f %f %f %f", tableunpack(coords))
3172 else
3173     err("unknown fading method '%s'", fd_type)
3174 end
3175 fd_type = fd_type == "linear" and 2 or 3
3176 local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3177 local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":"" }
3178 local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":"" }
3179 if steps > 1 then
3180     fractions = { prescript.sh_fraction_1 or 0 }
3181     for i=2,steps do
3182         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3183         ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":""
3184         cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":""
3185     end
3186 end
3187
3188 local pen = mplib.pen_info(object)
3189 if pen and pen.width then
3190     local wd = pen.width / 2
3191     bbox = { bbox[1]-wd, bbox[2]-wd, bbox[3]+wd, bbox[4]+wd }
3192 end
3193
3194 local dx, dy = -bbox[1], -bbox[2]
3195 local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3196 matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3197 matrix[5] = matrix[5] + dx
3198 matrix[6] = matrix[6] + dy
3199 matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3200 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3201 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3202 on = update_pdfobjs(os)
3203 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3204 local streamtext = format("q /Pattern cs/MPlibFd%s scn %s re f Q", on, bbox)
3205 :gsub(decimals,rmzeros)
3206 os = format("<</Pattern<</MPlibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3207 on = update_pdfobjs(os)
3208 local resources = format(pdfetcs.resfmt, on)
3209 on = update_pdfobjs"<</S/Transparency/CS/DeviceGray>>"
3210 local attr = tableconcat{
3211     "/Subtype/Form",
3212     "/BBox[" .. bbox .. "]",
3213     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
3214     "/Resources " .. resources,
3215     "/Group " .. format(pdfetcs.resfmt, on),
3216 } :gsub(decimals,rmzeros)

```

```

3217   on = update_pdfobjs(attr, streamtext)
3218   os = format("<</SMask<</S/Luminosity/G %s>>>", format(pdfetcs.resfmt, on))
3219   end
3220   on, new = update_pdfobjs(os)
3221   local key = add_extgs_resources(on,new)
3222   start_pdf_code()
3223   pdf_literalcode("/%s gs", key)
3224   if fd_stop then return "standalone" end
3225   return "start"
3226 end
3227

```

Transparency Group

```

3228 pdfetcs.tr_group = { shifts = { } }
3229 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3230 local function do_preobj_GRP (object, prescript)
3231   local grstate = prescript and prescript.gr_state
3232   if not grstate then return end
3233   local trgroup = pdfetcs.tr_group
3234   if grstate == "start" then
3235     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3236     trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3237     trgroup.wrapped = false
3238     for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+)") do
3239       trgroup[v] = true
3240     end
3241     trgroup.bbox = prescript.mplibgroupbbox:explode":"
3242     trgroup.widths = { }
3243     put2output[["\beginpgroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3244   elseif grstate == "stop" then
3245     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3246
3247     if #trgroup.widths > 0 then
3248       local wd = math.max(tableunpack(trgroup.widths))
3249       if wd and wd > 0 then
3250         wd = wd/2
3251         llx,lly,urx,ury = llx-wd, lly-wd, urx+wd, ury+wd
3252       end
3253     end
3254
3255     put2output(tableconcat{
3256       "\\egroup",
3257       format("\\wd\mplibscratchbox %fbp", urx-llx),
3258       format("\\ht\mplibscratchbox %fbp", ury-lly),
3259       "\\dp\mplibscratchbox 0pt",
3260     })
3261     local grattr
3262     if trgroup.off then
3263       grattr = ""

```

```

3264 else
3265     local on = update_pdfobjs(format("<</S/Transparency/I %s/K %s>>",
3266                                     trgroup.isolated, trgroup.knockout))
3267     grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3268 end
3269 local res = gather_resources("")
3270 local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3271 if pdfmode then
3272     if trgroup.wrapped then
3273         put2output(tableconcat{
3274             "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3275             "/BBox[" .. bbox .. "]} resources{" .. res .. "}" .. "\\mplibscratchbox",
3276             "\\setbox\\mplibscratchbox\\hbox{" .. useboxresource\\lastsavedboxresourceindex}",
3277         })
3278     end
3279     put2output(tableconcat{
3280         "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3281         "/BBox[" .. bbox .. "]", grattr, "} resources{" .. res .. "}" .. "\\mplibscratchbox",
3282         "\\luamplibtagasgroupput{" .. trgroup.name .. "}" ..,
3283         [[\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}]],
3284         [[\\wd\\mplibscratchbox 0pt\\ht\\mplibscratchbox 0pt\\dp\\mplibscratchbox 0pt]],
3285         [[\\box\\mplibscratchbox]],
3286         "\\endgroup",
3287         "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{" ..,
3288         "\\setbox\\mplibscratchbox\\hbox{\\hskip", -llx, "bp\\raise", -lly, "bp\\hbox{" ..,
3289         "\\useboxresource \\the\\lastsavedboxresourceindex",
3290         "}}\\wd\\mplibscratchbox", urx-llx, "bp\\ht\\mplibscratchbox", ury-lly, "bp",
3291         "\\box\\mplibscratchbox}",
3292     })
3293 else
3294     if trgroup.wrapped then
3295         trgroup.cnt = (trgroup.cnt or 0) + 1
3296         local objname = format("@mplibtrgr%s", trgroup.cnt)
3297         put2output(tableconcat{
3298             "\\special{pdf:bxobj " .. objname .. " bbox " .. bbox .. "}",
3299             "\\unhbox\\mplibscratchbox",
3300             "\\special{pdf:put @resources <<", res, ">>}",
3301             "\\special{pdf:exobj}",
3302             "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj " .. objname .. "}}",
3303         })
3304     end
3305     trgroup.cnt = (trgroup.cnt or 0) + 1
3306     local objname = format("@mplibtrgr%s", trgroup.cnt)
3307     put2output(tableconcat{
3308         "\\special{pdf:bxobj " .. objname .. " bbox " .. bbox .. "}",
3309         "\\unhbox\\mplibscratchbox",
3310         "\\special{pdf:put @resources <<", res, ">>}",
3311         "\\special{pdf:exobj <<", grattr, ">>}",
3312         "\\luamplibtagasgroupput{" .. trgroup.name .. "}" ..,

```

```

3313     "\\special{pdf:uxobj ", objname, "}",
3314     "}\endgroup",
3315     })
3316     token.set_macro("luamplib.group"..trgroup.name, tableconcat{
3317         "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3318         "\\special{pdf:uxobj ", objname, "}",
3319         "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3320         "\\box\\mplibscratchbox",
3321         }, "global")
3322     end
3323     trgroup.shifts[trgroup.name] = { llx, lly }
3324 end
3325 return grstate
3326 end
3327 function luamplib.registergroup (boxid, name, opts)
3328 if opts.asgroup and opts.asgroup:find"wrapped" then
3329     luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3330     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
3331     opts.asgroup = opts.asgroup:gsub("wrapped","")
3332 end
3333 local box = texgetbox(boxid)
3334 local wd, ht, dp = node.getwhd(box)
3335 local is_mask = opts.asgroup and opts.asgroup:find"masking"
3336 local res = opts.resources or ""
3337 res = gather_resources(res)
3338 local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3339 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
3340 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
3341 if opts.matrix and opts.matrix:find"%a" then
3342     local t = get_mp_matrix(opts.matrix)
3343     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3344 end
3345 local grtype = 3
3346 if opts.bbox then
3347     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3348     grtype = 2
3349 end
3350 local mllx, mply = get_macro'MPllx', get_macro'MPly'
3351 if is_mask then
3352     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3353     t[5], t[6] = t[5]+mllx, t[6]+mply
3354     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3355     mllx, mply = 0, 0
3356 end
3357 if opts.matrix then
3358     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3359     grtype = opts.bbox and 4 or 1
3360 end
3361 if opts.asgroup and not opts.asgroup:find"off" then

```

```

3362 local t = { isolated = false, knockout = false, masking = false }
3363 for _,v in ipairs(opts.asgroup:gsub("%s",""):explode",+")) do t[v] = true end
3364 local on
3365 if t.masking then
3366   on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorsname or "/DeviceGray"))
3367 else
3368   local cs = opts.colorsname and ("/CS%s"):format(opts.colorsname) or ""
3369   on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3370 end
3371 attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3372 end
3373 local trgroup = pdfetcs.tr_group
3374 trgroup.shifts[name] = { mply, mply }
3375 local whd
3376 if pdfmode then
3377   attr = tableconcat(attr) :gsub(decimals,rmzeros)
3378   local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3379   token.set_macro("luamplib.group"..name, tableconcat{
3380     "\\useboxresource ", index,
3381   }, "global")
3382   whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3383 else
3384   trgroup.cnt = (trgroup.cnt or 0) + 1
3385   local objname = format("@mplibtrgr%s", trgroup.cnt)
3386   texsprint {
3387     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3388     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3389     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3390     "\\special{pdf:bcontent}",
3391     "\\special{pdf:boxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3392     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3393     "\\special{pdf:put @resources <<", res, ">>}",
3394     "\\special{pdf:boxobj <<", tableconcat(attr), ">>}",
3395     "\\special{pdf:econtent}}",
3396   }
3397   token.set_macro("luamplib.group"..name, tableconcat{
3398     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:boxobj ", objname, "}}",
3399     "\\wd\\mplibscratchbox ", wd, "sp",
3400     "\\ht\\mplibscratchbox ", ht, "sp",
3401     "\\dp\\mplibscratchbox ", dp, "sp",
3402     "\\box\\mplibscratchbox",
3403   }, "global")
3404   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3405 end
3406 info("w/h/d of group '%s': %s", name, whd)
3407 end
3408

```

luamplib.convert: flushing figures

```

3409 do
3410   local function stop_special_effects(fade,opaq)
3411     if fade then -- fading
3412       stop_pdf_code()
3413     end
3414     if opaq then -- opacity
3415       pdf_literalcode(opaq)
3416     end
3417   end
3418 end

```

For parsing prescript materials.

```

3419 local function script2table(s)
3420   local t = {}
3421   for _,i in ipairs(s:explode("\13+")) do
3422     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3423     if k and v and k ~= "" and not t[k] then
3424       t[k] = v
3425     end
3426   end
3427   return t
3428 end
3429

```

For path shading with transformed pen

```

3430 local function invert_matrix (t)
3431   local a, b, c, d, x, y = t[1], t[2], t[3], t[4], t[5], t[6]
3432   local det = a*d - b*c
3433   assert(det ~= 0, 'transformation is not invertible!')
3434   return format("%f %f %f %f %f %f cm ",
3435     d/det, 0-b/det, 0-c/det, a/det, (d*x-b*y)/det, (a*y-c*x)/det)
3436 end
3437

```

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

```

3438 local function pdf_textfigure(font,size,text,width,height,depth)
3439   text = text:gsub(".",function(c)
3440     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
3441   end)
3442   put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
3443 end
3444
3445 local bend_tolerance = 131/65536
3446
3447 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3448
3449 local function pen_characteristics(object)
3450   local t = mplib.pen_info(object)
3451   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty

```

```

3452     divider = sx*sy - rx*ry
3453     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3454 end
3455
3456 local function concat(px, py) -- no tx, ty here
3457     return (sy*px-ry*py)/divider, (sx*py-rx*px)/divider
3458 end
3459
3460 local function curved(ith,pth)
3461     local d = pth.left_x - ith.right_x
3462     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3463        abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3464         d = pth.left_y - ith.right_y
3465         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3466            abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3467             return false
3468         end
3469     end
3470     return true
3471 end
3472
3473 local function flushnormalpath(path,open)
3474     local pth, ith
3475     for i=1,#path do
3476         pth = path[i]
3477         if not ith then
3478             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3479         elseif curved(ith,pth) then
3480             pdf_literalcode("%f %f %f %f %f c",
3481                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3482         else
3483             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3484         end
3485         ith = pth
3486     end
3487     if not open then
3488         local one = path[1]
3489         if curved(pth,one) then
3490             pdf_literalcode("%f %f %f %f %f c",
3491                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3492         else
3493             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3494         end
3495     elseif #path == 1 then -- special case .. draw point
3496         local one = path[1]
3497         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3498     end
3499 end
3500

```

```

3501 local function flushconcatpath(path,open)
3502   pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3503   local pth, ith
3504   for i=1,#path do
3505     pth = path[i]
3506     if not ith then
3507       pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3508     elseif curved(ith,pth) then
3509       local a, b = concat(ith.right_x,ith.right_y)
3510       local c, d = concat(pth.left_x,pth.left_y)
3511       pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3512     else
3513       pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3514     end
3515     ith = pth
3516   end
3517   if not open then
3518     local one = path[1]
3519     if curved(pth,one) then
3520       local a, b = concat(pth.right_x,pth.right_y)
3521       local c, d = concat(one.left_x,one.left_y)
3522       pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3523     else
3524       pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3525     end
3526   elseif #path == 1 then -- special case .. draw point
3527     local one = path[1]
3528     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3529   end
3530 end
3531

```

Finally, flush figures by inserting PDF literals.

```

3532 local function flush (result,flusher)
3533   if result then
3534     local figures = result.fig
3535     if figures then
3536       for f=1, #figures do
3537         info("flushing figure %s",f)
3538         local figure = figures[f]
3539         local objects = figure:objects()
3540         local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3541         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3542         local bbox = figure:boundingbox()
3543         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3544         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()
```

```
3545         else
```

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

```
3546         if tex_code_pre_mplib[f] then
3547             put2output(tex_code_pre_mplib[f])
3548         end
3549         pdf_startfigure(fignum,llx,lly,urx,ury)
3550         start_pdf_code()
3551         if objects then
3552             local savedpath = nil
3553             local savedhtap = nil
3554             for o=1,#objects do
3555                 local object      = objects[o]
3556                 local objecttype  = object.type
```

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

```
3557             local prescript      = object.prescript
3558             prescript = prescript and script2table(prescript) -- prescript is now a table
3559             do_preobj_CR(object,prescript) -- color
3560             local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3561             local fading_ = do_preobj_FADE(object,prescript) -- fading
3562             do_preobj_PAT(object,prescript) -- tiling pattern
3563             do_preobj_shading(object,prescript) -- shading pattern
3564             local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3565             if prescript and prescript.mplibtexboxid then
3566                 put_tex_boxes(object,prescript)
3567             elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3568             elseif objecttype == "start_clip" then
3569                 local evenodd = not object.istext and object.postscript == "evenodd"
3570                 start_pdf_code()
3571                 flushnormalpath(object.path,false)
3572                 pdf_literalcode(evenodd and "W* n" or "W n")
3573             elseif objecttype == "stop_clip" then
3574                 stop_pdf_code()
3575                 miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3576             elseif objecttype == "special" then
```

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

```
3577             if prescript and prescript.postmplibverbtex then
3578                 figcontents.post[#figcontents.post+1] = prescript.postmplibverbtex
3579             end
3580             elseif objecttype == "text" then
3581                 local ot = object.transform -- 3,4,5,6,1,2
3582                 start_pdf_code()
3583                 pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3584                 pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3585                 stop_pdf_code()
```

```

3586 elseif not trgroup and fading_ ~= "stop" then
3587   local evenodd, collect, both = false, false, false
3588   local postscript = object.postscript
3589   if not object.istext then
3590     if postscript == "evenodd" then
3591       evenodd = true
3592     elseif postscript == "collect" then
3593       collect = true
3594     elseif postscript == "both" then
3595       both = true
3596     elseif postscript == "eoboth" then
3597       evenodd = true
3598       both = true
3599     end
3600   end
3601   if collect then
3602     if not savedpath then
3603       savedpath = { object.path or false }
3604       savedhtap = { object.htap or false }
3605     else
3606       savedpath[#savedpath+1] = object.path or false
3607       savedhtap[#savedhtap+1] = object.htap or false
3608     end
3609   else

```

Removed from ConT_EXt general: color stuff.

```

3610   local ml = object.miterlimit
3611   if ml and ml ~= miterlimit then
3612     miterlimit = ml
3613     pdf_literalcode("%f M",ml)
3614   end
3615   local lj = object.linejoin
3616   if lj and lj ~= linejoin then
3617     linejoin = lj
3618     pdf_literalcode("%i j",lj)
3619   end
3620   local lc = object.linecap
3621   if lc and lc ~= linecap then
3622     linecap = lc
3623     pdf_literalcode("%i J",lc)
3624   end
3625   local dl = object.dash
3626   if dl then
3627     local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3628     if d ~= dashed then
3629       dashed = d
3630       pdf_literalcode(dashed)
3631     end
3632   elseif dashed then

```

```

3633         pdf_literalcode("[[] 0 d")
3634         dashed = false
3635     end
3636     local path = object.path
3637     local transformed, penwidth = false, 1
3638     local open = path and path[1].left_type and path[#path].right_type
3639     local pen = object.pen
3640     if pen then
3641         if pen.type == 'elliptical' then
3642             transformed, penwidth = pen_characteristics(object) -- boolean, value

```

next three lines inserted for asgroup objects drawn with a thick pen

```

3643         if penwidth and pdfetcs.tr_group.widths then
3644             tableinsert(pdfetcs.tr_group.widths, penwidth)
3645         end
3646         pdf_literalcode("%f w",penwidth)
3647         if objecttype == 'fill' then
3648             objecttype = 'both'
3649         end
3650         else -- calculated by mplib itself
3651             objecttype = 'fill'
3652         end
3653     end

```

Added : shading

```

3654     local shade_no, shade_stroke, shade_cm, shade_im = do_preobj_SH(object,prescript) -- shading
3655     if shade_no then
3656         objecttype = "shade"
3657         shade_im = transformed and invert_matrix{sx, rx, ry, sy, tx ,ty} or ""
3658         shade_cm = shade_cm and shade_cm.." cm " or ""
3659     end
3660     if transformed then
3661         start_pdf_code()
3662     end
3663     if path then
3664         if savedpath then
3665             for i=1,#savedpath do
3666                 local path = savedpath[i]
3667                 if transformed then
3668                     flushconcatpath(path,open)
3669                 else
3670                     flushnormalpath(path,open)
3671                 end
3672             end
3673             savedpath = nil
3674         end
3675         if transformed then
3676             flushconcatpath(path,open)
3677         else
3678             flushnormalpath(path,open)

```

```

3679         end
3680         if objecttype == "fill" then
3681             pdf_literalcode(evenodd and "h f*" or "h f")
3682         elseif objecttype == "outline" then
3683             if both then
3684                 pdf_literalcode(evenodd and "h B*" or "h B")
3685             else
3686                 pdf_literalcode(open and "S" or "h S")
3687             end
3688         elseif objecttype == "both" then
3689             pdf_literalcode(evenodd and "h B*" or "h B")
3690         elseif objecttype == "shade" then
3691             pdf_literalcode("q W%s %s %s%s/MPlibSh%s sh Q",
3692                 evenodd and "*" or "",
3693                 shade_stroke and "s" or "n",
3694                 shade_im, shade_cm, shade_no)
3695         end
3696     end
3697     if transformed then
3698         stop_pdf_code()
3699     end
3700     local path = object.htap

```

htap objects are generated by, say, filldraw with pensquare.

```

3701     if path then
3702         if transformed then
3703             start_pdf_code()
3704         end
3705         if savedhtap then
3706             for i=1,#savedhtap do
3707                 local path = savedhtap[i]
3708                 if transformed then
3709                     flushconcatpath(path,open)
3710                 else
3711                     flushnormalpath(path,open)
3712                 end
3713             end
3714             savedhtap = nil
3715             evenodd = true
3716         end
3717         if transformed then
3718             flushconcatpath(path,open)
3719         else
3720             flushnormalpath(path,open)
3721         end
3722         if objecttype == "fill" then
3723             pdf_literalcode(evenodd and "h f*" or "h f")
3724         elseif objecttype == "outline" then
3725             pdf_literalcode(open and "S" or "h S")

```

```

3726         elseif objecttype == "both" then
3727             pdf_literalcode(evenodd and "h B*" or "h B")
3728         elseif objecttype == "shade" then
3729             pdf_literalcode("q W%s %s %s%/MPlibSh%s sh Q",
3730                 evenodd and "*" or "",
3731                 shade_stroke and "s" or "n",
3732                 shade_im, shade_cm, shade_no)
3733         end
3734         if transformed then
3735             stop_pdf_code()
3736         end
3737     end

```

Added to ConT_EXt general: post-object special effects. Beware q ... Q scope.

```

3738         end
3739     end
3740     if fading_ == "start" then
3741         pdfetcs.fading.specialeffects = {fading_, tr_opaq}
3742     elseif trgroup == "start" then
3743         pdfetcs.tr_group.specialeffects = {fading_, tr_opaq}
3744     elseif fading_ == "stop" then
3745         local se = pdfetcs.fading.specialeffects
3746         if se then stop_special_effects(se[1], se[2]) end
3747     elseif trgroup == "stop" then
3748         local se = pdfetcs.tr_group.specialeffects
3749         if se then stop_special_effects(se[1], se[2]) end
3750     else
3751         stop_special_effects(fading_, tr_opaq)
3752     end
3753     if fading_ or trgroup then -- extgs reseted
3754         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3755     end
3756 end
3757 end
3758 stop_pdf_code()
3759 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimt_Ex code.

```

3760     for _,v in ipairs(figcontents) do
3761         if type(v) == "table" then
3762             texsprint("\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint("}")
3763         else
3764             texsprint(v)
3765         end
3766     end
3767     if #figcontents.post > 0 then texsprint(figcontents.post) end
3768     figcontents = { post = { } }
3769 end
3770 end
3771 end

```

```

3772   end
3773 end
3774
3775 function luamplib.convert (result, flusher)
3776   flush(result, flusher)
3777   return true -- done
3778 end
3779 end
3780
3781 function luamplib.colorconverter (cr)
3782   local n = #cr
3783   if n == 4 then
3784     local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3785     return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K", c,m,y,k,c,m,y,k), "0 g 0 G"
3786   elseif n == 3 then
3787     local r, g, b = cr[1], cr[2], cr[3]
3788     return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG", r,g,b,r,g,b), "0 g 0 G"
3789   else
3790     local s = cr[1]
3791     return format("%.3f g %.3f G", s,s), "0 g 0 G"
3792   end
3793 end

```

2.2 $\TeX$ package

First we need to load some packages.

```

3794 \ifcsname ProvidesPackage\endcsname

```

We need $\TeX$ 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.

```

3795 \NeedsTeXFormat{LaTeX2e}
3796 \ProvidesPackage{luamplib}
3797   [2026/09/01 v2.42.9 mplib package for LuaTeX]
3798 \fi
3799 \ifdefined\newluafunction\else
3800   \input ltluatex
3801 \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 $\TeX$ kernel. In Plain, `atbegshi.sty` is loaded.

```

3802 \ifnum\outputmode=0
3803   \ifdefined\AddToHookNext
3804     \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3805     \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3806     \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3807   \else
3808     \input atbegshi.sty
3809     \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}

```

```

3810 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3811 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3812 \fi
3813 \fi

```

Loading of lua code.

```

3814 \directlua{require("luamplib")}

```

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

```

3815 \ifx\pdfoutput\undefined
3816 \let\pdfoutput\outputmode
3817 \fi
3818 \ifx\pdfliteral\undefined
3819 \protected\def\pdfliteral{\pdfextension literal}
3820 \fi

```

Set the format for METAPOST.

```

3821 \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.

```

3822 \ifnum\pdfoutput>0
3823 \let\mplibtoPDF\pdfliteral
3824 \else
3825 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3826 \ifcsname PackageInfo\endcsname
3827 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3828 \else
3829 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3830 \fi
3831 \fi

```

To make mplibcode typeset always in horizontal mode.

```

3832 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3833 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3834 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

3835 \def\mplibsetupcatcodes{%
3836 %catcode`\{=12 %catcode`\}=12
3837 \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3838 \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
3839 }

```

Make btex...etex box zero-metric. Plus address the issue #189. bdir 0 seems to be redundant, but no harm.

```

3840 \ifnum\outputmode>0 %
3841 \def\mplibputtextbox#1#2#3#4{%
3842 \pdfextension save\relax
3843 \vbox to 0pt{\vss
3844 \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
3845 \kern#3bp}%

```

```

3846 \pdfextension restore\relax}
3847 \else
3848 \def\mplibputtextbox#1#2#3#4{%
3849 \special{pdf:btrans matrix #4 #2 #3}%
3850 \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
3851 \special{pdf:etrans}}
3852 \fi

      use Transparency Group

3853 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3854 \def\usemplibgroupmain#1{%
3855 \prependtomplibbox\hbox dir TLT\bgroup
3856 \csname luamplib.group.#1\endcsname
3857 \egroup
3858 }
3859 \protected\def\mplibgroup#1{%
3860 \begingroup
3861 \def\MPllx{0}\def\MPlly{0}%
3862 \def\mplibgroupname{#1}%
3863 \mplibgroupgetnexttok
3864 }
3865 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3866 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
3867 \def\mplibgroupbranch{%
3868 \ifx [\nexttok
3869 \expandafter\mplibgroupopts
3870 \else
3871 \ifx\mplibsptoken\nexttok
3872 \expandafter\expandafter\expandafter\mplibgroupskipspace
3873 \else
3874 \let\mplibgroupoptions\empty
3875 \expandafter\expandafter\expandafter\mplibgroupmain
3876 \fi
3877 \fi
3878 }
3879 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3880 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3881 \protected\def\endmplibgroup{\egroup
3882 \directlua{ luamplib.registergroup(
3883 \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3884 )}}%
3885 \endgroup
3886 }

      Patterns

3887 {\def\:\global\let\mplibsptoken= }\: }
3888 \protected\def\mppattern#1{%
3889 \begingroup
3890 \def\MPllx{0}\def\MPlly{0}%
3891 \def\mplibpatternname{#1}%

```

```

3892 \mplibpatterngetnexttok
3893 }
3894 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3895 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3896 \def\mplibpatternbranch{%
3897   \ifx [\nexttok
3898     \expandafter\mplibpatternopts
3899   \else
3900     \ifx\mplibsptoken\nexttok
3901       \expandafter\expandafter\expandafter\mplibpatternskipsspace
3902     \else
3903       \let\mplibpatternoptions\empty
3904       \expandafter\expandafter\expandafter\mplibpatternmain
3905     \fi
3906   \fi
3907 }
3908 \def\mplibpatternopts[#1]{%
3909   \def\mplibpatternoptions{#1}%
3910   \mplibpatternmain
3911 }
3912 \def\mplibpatternmain{%
3913   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3914 }
3915 \protected\def\endmppattern{%
3916   \egroup
3917   \directlua{ luamplib.registerpattern(
3918     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3919   )}%
3920   \endgroup
3921 }

```

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

```

3922 \def\mpfiginstancename{@mpfig}
3923 \protected\def\mpfig{%
3924   \begingroup
3925   \futurelet\nexttok\mplibmpfigbranch
3926 }
3927 \def\mplibmpfigbranch{%
3928   \ifx *\nexttok
3929     \expandafter\mplibprempfig
3930   \else
3931     \ifx [\nexttok
3932       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3933     \else
3934       \expandafter\expandafter\expandafter\mplibmainmpfig
3935     \fi
3936   \fi
3937 }
3938 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}

```

```

3939 \def\mplibmainmpfig{%
3940   \begingroup
3941   \mplibsetupcatcodes
3942   \mplibdomainmpfig
3943 }
3944 \long\def\mplibdomainmpfig#1\endmpfig{%
3945   \endgroup
3946   \directlua{
3947     local legacy = luamplib.legacyverbatim
3948     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3949     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3950     luamplib.legacyverbatim = false
3951     luamplib.everymplib["\mpfiginstancename"] = ""
3952     luamplib.everyendmplib["\mpfiginstancename"] = ""
3953     luamplib.process_mplibcode(
3954       "beginfig(0) "..everympfig.." "..[==[\unexpanded{#1}]===].." "..everyendmpfig.." endfig;",
3955       "\mpfiginstancename")
3956     luamplib.legacyverbatim = legacy
3957     luamplib.everymplib["\mpfiginstancename"] = everympfig
3958     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3959   }%
3960   \endgroup
3961 }
3962 \def\mplibprempfig#1{%
3963   \begingroup
3964   \mplibsetupcatcodes
3965   \mplibdoprempfig
3966 }
3967 \long\def\mplibdoprempfig#1\endmpfig{%
3968   \endgroup
3969   \directlua{
3970     local legacy = luamplib.legacyverbatim
3971     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3972     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3973     luamplib.legacyverbatim = false
3974     luamplib.everymplib["\mpfiginstancename"] = ""
3975     luamplib.everyendmplib["\mpfiginstancename"] = ""
3976     luamplib.process_mplibcode([==[\unexpanded{#1}]===], "\mpfiginstancename")
3977     luamplib.legacyverbatim = legacy
3978     luamplib.everymplib["\mpfiginstancename"] = everympfig
3979     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3980   }%
3981   \endgroup
3982 }
3983 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3984 \unless\ifcsname ver@luamplib.sty\endcsname
3985   \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}

```

```

3986 \protected\def\mplibcode{%
3987   \begingroup
3988   \futurelet\nexttok\mplibcodebranch
3989 }
3990 \def\mplibcodebranch{%
3991   \ifx [\nexttok
3992     \expandafter\mplibcodegetinstancename
3993   \else
3994     \global\let\currentmpinstancename\empty
3995     \expandafter\mplibcodeindeed
3996   \fi
3997 }
3998 \def\mplibcodeindeed{%
3999   \begingroup
4000   \mplibsetupcatcodes
4001   \mplibdocode
4002 }
4003 \long\def\mplibdocode#1\endmplibcode{%
4004   \endgroup
4005   \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]===],"currentmpinstancename")}%
4006   \endgroup
4007 }
4008 \protected\def\endmplibcode{endmplibcode}
4009 \else

```

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

```

4010 \newenvironment{mplibcode}[1][{}]{%
4011   \xdef\currentmpinstancename{#1}%
4012   \mplibtmp toks{}\ltxdomplibcode
4013 }{}
4014 \def\ltxdomplibcode{%
4015   \begingroup
4016   \mplibsetupcatcodes
4017   \ltxdomplibcodeindeed
4018 }
4019 \def\mplib@mplibcode{mplibcode}
4020 \long\def\ltxdomplibcodeindeed#1\end#2{%
4021   \endgroup
4022   \mplibtmp toks\expandafter{\the\mplibtmp toks#1}%
4023   \def\mplibtemp@a{#2}%
4024   \ifx\mplib@mplibcode\mplibtemp@a
4025     \directlua{luamplib.process_mplibcode([==[\the\mplibtmp toks]===],"currentmpinstancename")}%
4026     \end{mplibcode}%
4027   \else
4028     \mplibtmp toks\expandafter{\the\mplibtmp toks\end{#2}}%
4029     \expandafter\ltxdomplibcode
4030   \fi
4031 }
4032 \fi

```

User settings.

```

4033 \def\mplibshowlog#1{\directlua{
4034   local s = string.lower("#1")
4035   if s == "enable" or s == "true" or s == "yes" then
4036     luamplib.showlog = true
4037   else
4038     luamplib.showlog = false
4039   end
4040 }}
4041 \def\mpliblegacybehavior#1{\directlua{
4042   local s = string.lower("#1")
4043   if s == "enable" or s == "true" or s == "yes" then
4044     luamplib.legacyverbatim = true
4045   else
4046     luamplib.legacyverbatim = false
4047   end
4048 }}
4049 \def\mplibverbatim#1{\directlua{
4050   local s = string.lower("#1")
4051   if s == "enable" or s == "true" or s == "yes" then
4052     luamplib.verbatiminput = true
4053   else
4054     luamplib.verbatiminput = false
4055   end
4056 }}
4057 \newtoks\mplibtmptoks

```

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

```

4058 \ifcsname ver@luamplib.sty\endcsname
4059   \protected\def\everymplib{%
4060     \begingroup
4061     \mplibsetupcatcodes
4062     \mplibdoeverymplib
4063   }
4064   \protected\def\everyendmplib{%
4065     \begingroup
4066     \mplibsetupcatcodes
4067     \mplibdoeveryendmplib
4068   }
4069   \newcommand\mplibdoeverymplib[2][{}]{%
4070     \endgroup
4071     \directlua{
4072       luamplib.everymplib["#1"] = [==[\unexpanded{#2}]==]
4073     }%
4074   }
4075   \newcommand\mplibdoeveryendmplib[2][{}]{%
4076     \endgroup
4077     \directlua{
4078       luamplib.everyendmplib["#1"] = [==[\unexpanded{#2}]==]

```

```

4079 }%
4080 }
4081 \else
4082 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
4083 \protected\def\everymplib#1{%
4084 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4085 \begingroup
4086 \mplibsetupcatcodes
4087 \mplibdoeverymplib
4088 }
4089 \long\def\mplibdoeverymplib#1{%
4090 \endgroup
4091 \directlua{
4092 \luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
4093 }%
4094 }
4095 \protected\def\everyendmplib#1{%
4096 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4097 \begingroup
4098 \mplibsetupcatcodes
4099 \mplibdoeveryendmplib
4100 }
4101 \long\def\mplibdoeveryendmplib#1{%
4102 \endgroup
4103 \directlua{
4104 \luamplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
4105 }%
4106 }
4107 \fi

```

TeX macros for dimen/color

```

4108 \def\mpdim#1{ \runscript{"luamplibdimen{#1}"} }
4109 \def\mpcolor#1#{\domplibcolor{#1}}
4110 \def\domplibcolor#1#2{ \runscript{"luamplibcolor{#1}{#2}"} }

```

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

```

4111 \def\mplibnumbersystem#1{\directlua{
4112 local t = "#1"
4113 if t == "binary" then t = "decimal" end
4114 \luamplib.numbersystem = t
4115 }}

```

Settings for .mp cache files.

```

4116 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4117 \def\mplibdomakenocache#1,{%
4118 \ifx\empty#1\empty
4119 \expandafter\mplibdomakenocache
4120 \else
4121 \ifx\stop#1\else
4122 \directlua{\luamplib.noneedtoreplace["#1.mp"]=true}%

```

```

4123     \expandafter\expandafter\expandafter\mplibdomakenocache
4124     \fi
4125     \fi
4126 }
4127 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4128 \def\mplibdocancelnocache#1,{%
4129     \ifx\empty#1\empty
4130         \expandafter\mplibdocancelnocache
4131     \else
4132         \ifx\stop#1\else
4133             \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
4134             \expandafter\expandafter\expandafter\mplibdocancelnocache
4135         \fi
4136     \fi
4137 }
4138 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4139 \def\mplibtexttextlabel#1{\directlua{
4140     local s = string.lower("#1")
4141     if s == "enable" or s == "true" or s == "yes" then
4142         luamplib.texttextlabel = true
4143     else
4144         luamplib.texttextlabel = false
4145     end
4146 }}
4147 \def\mplibcodeinherit#1{\directlua{
4148     local s = string.lower("#1")
4149     if s == "enable" or s == "true" or s == "yes" then
4150         luamplib.codeinherit = true
4151     else
4152         luamplib.codeinherit = false
4153     end
4154 }}
4155 \def\mplibglobaltexttext#1{\directlua{
4156     local s = string.lower("#1")
4157     if s == "enable" or s == "true" or s == "yes" then
4158         luamplib.globaltexttext = true
4159     else
4160         luamplib.globaltexttext = false
4161     end
4162 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

4163 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4164 \def\mplibstarttoPDF#1#2#3#4{%
4165     \prependtomplibbox

```

```

4166 \hbox dir TLT\bgroup
4167 \xdef\MPllx{#1}\xdef\MPlly{#2}%
4168 \xdef\MPurx{#3}\xdef\MPury{#4}%
4169 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4170 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4171 \parskip0pt%
4172 \leftskip0pt%
4173 \parindent0pt%
4174 \everypar{}%
4175 \setbox\mplibscratchbox\vbox\bgroup
4176 \noindent
4177 }
4178 \def\mplibstoptoPDF{%
4179 \par
4180 \egroup %
4181 \setbox\mplibscratchbox\hbox %
4182 {\hskip-\MPllx bp%
4183 \raise-\MPlly bp%
4184 \box\mplibscratchbox}%
4185 \setbox\mplibscratchbox\vbox to \MPheight
4186 {\vfill
4187 \hsize\MPwidth
4188 \wd\mplibscratchbox0pt%
4189 \ht\mplibscratchbox0pt%
4190 \dp\mplibscratchbox0pt%
4191 \box\mplibscratchbox}%
4192 \wd\mplibscratchbox\MPwidth
4193 \ht\mplibscratchbox\MPheight
4194 \box\mplibscratchbox
4195 \egroup
4196 }

```

Text items have a special handler.

```

4197 \def\mplibtexttext#1#2#3#4#5{%
4198 \begingroup
4199 \setbox\mplibscratchbox\hbox
4200 {\font\temp=#1 at #2bp%
4201 \temp
4202 #3}%
4203 \setbox\mplibscratchbox\hbox
4204 {\hskip#4 bp%
4205 \raise#5 bp%
4206 \box\mplibscratchbox}%
4207 \wd\mplibscratchbox0pt%
4208 \ht\mplibscratchbox0pt%
4209 \dp\mplibscratchbox0pt%
4210 \box\mplibscratchbox
4211 \endgroup
4212 }

```

Input luamplib.cfg when it exists.

```
4213 \openin0=luamplib.cfg
4214 \ifeof0 \else
4215   \closein0
4216   \input luamplib.cfg
4217 \fi
```

Code for tagpdf

```
4218 \def\luamplibtagtextboxset#1#2{#2}
4219 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4220 \let\luamplibtagasgroupset\relax
4221 \let\luamplibtagasgroupput\luamplibtagtextboxset
4222 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4223 \ifcsname ver@tagpdf.sty\endcsname \else
4224   \ExplSyntaxOn
4225   \keys_define:nn{luamplib/tagging}
4226   {
4227     ,alt          .code:n = { }
4228     ,actualtext   .code:n = { }
4229     ,artifact     .code:n = { }
4230     ,text         .code:n = { }
4231     ,off          .code:n = { }
4232     ,tag          .code:n = { }
4233     ,adjust-BBox  .code:n = { }
4234     ,tagging-setup .code:n = { }
4235     ,instance     .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4236     ,instancename .meta:n = { instance = {#1} }
4237     ,unknown      .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4238   }
4239   \RenewDocumentCommand\mplibcode{O{}}
4240   {
4241     \tl_gclear:N \currentmpinstancename
4242     \keys_set:ne{luamplib/tagging}{#1}
4243     \mplibtmptoks{}\ltxdomplibcode
4244   }
4245   \cs_set_eq:NN \mplibaltext \use_none:n
4246   \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.

```
4247 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4248 \ExplSyntaxOff
4249 \endinput\fi
4250 \ExplSyntaxOn
4251 \tl_new:N \l__luamplib_tag_envname_tl
4252 \tl_new:N \l__luamplib_tag_alt_tl
```

```

4253 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4254 \tl_new:N \l__luamplib_tag_actual_tl
4255 \tl_new:N \l__luamplib_tag_struct_tl
4256 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4257 \bool_new:N \l__luamplib_tag_usetext_bool
4258 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4259 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4260 \tl_new:N \l__luamplib_tag_bbox_draw_tl
4261 \tl_new:N \l__luamplib_BBox_llx_tl
4262 \tl_new:N \l__luamplib_BBox_lly_tl
4263 \tl_new:N \l__luamplib_BBox_urx_tl
4264 \tl_new:N \l__luamplib_BBox_ury_tl
4265 \msg_new:nnn {luamplib}{figure-text-reuse}
4266 {
4267   tex-text~box~#1~probably~is~incorrectly~tagged.~
4268   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4269   Check~the~resulting~PDF.
4270 }
4271 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4272 {
4273   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4274   Using~mplibgroup~with~text~mode~is~not~recommended.~
4275   Check~the~resulting~PDF.
4276 }
4277 \msg_new:nnn {luamplib}{alt-text-missing}
4278 {
4279   Alternate~text~for~#1~is~missing.~
4280   Using~the~default~value~'#2'~instead.
4281 }

```

Sockets for tex-text boxes.

```

4282 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4283 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4284 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4285 {

```

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.

```

4286 \bool_if:NTF \l__luamplib_tag_usetext_bool
4287 {
4288   \tag_mc_end_push:
4289   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4290   \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.

```

4291   \tag_mc_begin:n{tag=text}
4292   #2
4293   \tag_mc_end:
4294   \tag_struct_end:
4295   \tag_mc_begin_pop:n{ }

```

```

4296 }
4297 {
4298   \tag_suspend:n{\luamplibtagtextboxset}
4299   #2
4300   \tag_resume:n{\luamplibtagtextboxset}
4301 }
4302 }
4303 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4304 {
4305   \bool_lazy_and:nnTF
4306   { \l__luamplib_tag_usetext_bool }
4307   { \cs_if_free_p:c {luamplib.taggedbox.#1} }
4308   {
4309     \tag_resume:n{\mplibputtextbox}
4310     \tag_mc_end:
4311     \cs_if_exist:cTF {luamplib.taggedbox.#1}
4312     {
4313       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4314       #2
4315       \cs_undefine:c {luamplib.taggedbox.#1}
4316     }
4317     {
4318       \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
4319       \tag_mc_begin:n{ }
4320       \int_set:Nn \l_tmpa_int {#1}
4321       \tag_mc_reset_box:N \l_tmpa_int
4322       #2
4323       \tag_mc_end:
4324     }
4325     \tag_mc_begin:n{artifact}
4326   }
4327   {
4328     \int_set:Nn \l_tmpa_int {#1}
4329     \tag_mc_reset_box:N \l_tmpa_int
4330     #2
4331   }
4332 }
4333 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4334 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4335 \cs_set_nopar:Npn \luamplibtagtextboxset
4336 {
4337   \tag_socket_use:nnn{luamplib/texttext/set}
4338 }

```

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

```

4339 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4340 {
4341   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool

```

```

4342 \bool_set_false:N \l__luamplib_tag_usetext_bool
4343 \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
4344 \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4345 \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4346 }
4347 \sys_if_output_pdf:TF
4348 {
4349 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4350 {
4351 \pdfextension save\relax
4352 \vbox to 0pt{\vss
4353 \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4354 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4355 \kern #3bp}
4356 \pdfextension restore\relax
4357 }
4358 }
4359 {
4360 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4361 {
4362 \special{pdf:btrans~matrix~#4~#2~#3}
4363 \vbox to 0pt{\vss\hbox bdir0 to 0pt{
4364 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4365 \special{pdf:etrans}
4366 }
4367 }

```

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

```

4368 \cs_set_nopar:Npn \luamplibtagasgroupset
4369 {
4370 \bool_set_false:N \l__luamplib_tag_usetext_bool
4371 }
4372 \cs_set_nopar:Npn \luamplibtagasgroupput
4373 {
4374 \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4375 \tag_socket_use:nnn{luamplib/mplibgroup/put}
4376 }

```

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

```

4377 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4378 \socket_new_plugin:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4379 {
4380 \cs_if_free:cT {luamplib.mplibgroup.text.#1}
4381 {
4382 \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4383 \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4384 }
4385 \tag_mc_end:
4386 \tag_mc_begin:n{tag=text}

```

```

4387 #2
4388 \tag_mc_end:
4389 \tag_mc_begin:n{artifact}
4390 }
4391 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

A macro for BBox attribute

```

4392 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4393 {
4394   \tl_set:Nx \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4395   \tex_savepos:D
4396   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4397   \tl_set:Nx \l__luamplib_BBox_llx_tl
4398     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4399   \tl_set:Nx \l__luamplib_BBox_lly_tl
4400     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4401   \tl_set:Nx \l__luamplib_BBox_urx_tl
4402     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4403   \tl_set:Nx \l__luamplib_BBox_ury_tl
4404     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4405   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4406   {
4407     \int_zero:N \l_tmpa_int
4408     \tl_map_inline:nn
4409     {
4410       \l__luamplib_BBox_llx_tl
4411       \l__luamplib_BBox_lly_tl
4412       \l__luamplib_BBox_urx_tl
4413       \l__luamplib_BBox_ury_tl
4414     }
4415     {
4416       \int_incr:N \l_tmpa_int
4417       \tl_set:Nx ##1
4418       {
4419         \fp_eval:n
4420         {
4421           ##1
4422           +
4423           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4424         }
4425       }
4426     }
4427   }
4428   \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4429   {
4430     /O /Layout /BBox [
4431       \l__luamplib_BBox_llx_tl\c_space_tl
4432       \l__luamplib_BBox_lly_tl\c_space_tl
4433       \l__luamplib_BBox_urx_tl\c_space_tl

```

```

4434 \l__luamplib_BBox_ury_tl
4435 ]
4436 }
4437 \bool_if:NT \l__tag_graphic_debug_bool
4438 {
4439 \iow_log:e
4440 {
4441 \luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4442 \l__luamplib_BBox_llx_tl\c_space_tl
4443 \l__luamplib_BBox_lly_tl\c_space_tl
4444 \l__luamplib_BBox_urx_tl\c_space_tl
4445 \l__luamplib_BBox_ury_tl
4446 }
4447 \sys_if_output_pdf:TF
4448 {
4449 \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
4450 {
4451 \pdfextension save\relax
4452 \opacity_select:n{0.5} \color_select:n{red}
4453 \pdfextension literal~text
4454 {
4455 \l__luamplib_BBox_llx_tl\c_space_tl
4456 \l__luamplib_BBox_lly_tl\c_space_tl
4457 \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4458 \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4459 re~f
4460 }
4461 \pdfextension restore\relax
4462 }
4463 }
4464 {
4465 \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
4466 {
4467 \special{pdf:bcontent}
4468 \opacity_select:n{0.5} \color_select:n{red}
4469 \special{pdf:code~
4470 1~0~0~1~
4471 -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4472 -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4473 cm
4474 }
4475 \special{pdf:code~
4476 \l__luamplib_BBox_llx_tl\c_space_tl
4477 \l__luamplib_BBox_lly_tl\c_space_tl
4478 \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4479 \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4480 re~f
4481 }
4482 \special{pdf:econtent}

```

```

4483     }
4484   }
4485 }
4486 }

```

Sockets for main process

```

4487 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4488 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4489 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4490 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4491 {
4492   \tag_mc_end_push:
4493   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4494   {
4495     \tl_if_empty:eTF{#1}
4496     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4497     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4498     \msg_warning:nnVV{luamplib}{alt-text-missing}
4499     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4500   }
4501   \tag_struct_begin:n
4502   {
4503     tag=\l__luamplib_tag_struct_tl,
4504     alt=\l__luamplib_tag_alt_tl,
4505   }
4506   \tag_mc_begin:n{}
4507 }
4508 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4509 {
4510   \__luamplib_tag_bbox_attribute:n {#1}
4511   #2
4512   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4513   \tag_mc_end:
4514   \tag_struct_end:
4515   \tag_mc_begin_pop:n{}
4516 }
4517 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4518 {
4519   \tag_mc_end_push:
4520   \tag_struct_begin:n
4521   {
4522     tag=Span,
4523     actualtext=\l__luamplib_tag_actual_tl,
4524   }
4525   \tag_mc_begin:n{}
4526 }
4527 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4528 {
4529   #2

```

```

4530 \tag_mc_end:
4531 \tag_struct_end:
4532 \tag_mc_begin_pop:n{ }
4533 }
4534 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4535 {
4536 \tag_mc_end_push:
4537 \tag_mc_begin:n{artifact}
4538 }
4539 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4540 {
4541 #2
4542 \tag_mc_end:
4543 \tag_mc_begin_pop:n{ }
4544 }

```

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

```

4545 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4546 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4547 {
4548 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4549 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4550 }
4551 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4552 {
4553 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4554 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4555 \prependtomplibbox \mplibnoforcehmode
4556 \mode_if_vertical:T { \noindent \aftergroup\par }
4557 }
4558 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4559 {
4560 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4561 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4562 }
4563 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4564 {
4565 \bool_set_true:N \l__luamplib_tag_usetext_bool
4566 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4567 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4568 \prependtomplibbox \mplibnoforcehmode
4569 \mode_if_vertical:T { \noindent \aftergroup\par }
4570 }
4571 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4572 {
4573 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4574 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}

```

```

4575 }
4576 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

Key-value options
4577 \keys_define:nn{luamplib/tagging}
4578 {
4579   ,alt .code:n =
4580   {
4581     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4582     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4583   }
4584   ,actualtext .code:n =
4585   {
4586     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4587     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4588   }
4589   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4590   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4591   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4592   ,tag .code:n =
4593   {
4594     \str_case:nnF {#1}
4595     {
4596       {false} { \keys_set:nn {luamplib/tagging} {off} }
4597       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4598     }
4599     {
4600       \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4601       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4602     }
4603   }
4604   ,adjust-BBox .code:n =
4605   {
4606     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4607     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4608   }
4609   ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4610 }
4611 \keys_define:nn {luamplib/instance}
4612 {
4613   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4614   ,instancename .meta:n = { instance = {#1} }
4615   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4616 }

```

Redefine our macros

```

4617 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4618 {
4619   \prependtomplibbox
4620   \hbox dir~TLT\bgroup

```

```

4621 \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4622 \xdef\MPllx{#1}\xdef\MPlly{#2}%
4623 \xdef\MPurx{#3}\xdef\MPury{#4}%
4624 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4625 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4626 \parskip0pt
4627 \leftskip0pt
4628 \parindent0pt
4629 \everypar{}%
4630 \setbox\mplibscratchbox\vbox\bgroup
4631 \tag_suspend:n{\mplibstarttoPDF}
4632 \noindent
4633 }
4634 \cs_set_nopar:Npn \mplibstoptoPDF
4635 {
4636 \par
4637 \egroup
4638 \setbox\mplibscratchbox\hbox
4639 {\hskip-\MPllx bp
4640 \raise-\MPlly bp
4641 \box\mplibscratchbox}%
4642 \setbox\mplibscratchbox\vbox to \MPheight
4643 {\vfill
4644 \hsize\MPwidth
4645 \wd\mplibscratchbox0pt
4646 \ht\mplibscratchbox0pt
4647 \dp\mplibscratchbox0pt
4648 \box\mplibscratchbox}%
4649 \wd\mplibscratchbox\MPwidth
4650 \ht\mplibscratchbox\MPheight
4651 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4652 \egroup
4653 }
4654 \RenewDocumentCommand\mplibcode{0{}}
4655 {
4656 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4657 \tl_gclear:N \currentmpinstancename
4658 \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4659 \keys_set:nV {luamplib/instance} \l_tmpa_tl
4660 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4661 \tag_socket_use:n{luamplib/figure/init}
4662 \mplibtmptoks{}\ltxdomplibcode
4663 }
4664 \RenewDocumentCommand\mpfig{s 0{}}
4665 {
4666 \beginngroup
4667 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4668 \keys_set_known:ne {luamplib/tagging} {#2}
4669 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename

```

```

4670 \tag_socket_use:n{luamplib/figure/init}
4671 \IfBooleanTF{#1} { \mplibprempfig * }
4672           { \mplibmainmpfig }
4673 }
4674 \RenewDocumentCommand\usemplibgroup{0{} m}
4675 {
4676   \begingroup
4677   \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4678   \keys_set_known:ne {luamplib/tagging} {#1}
4679   \tag_socket_use:n{luamplib/figure/init}
4680   \prependtomplibbox\hbox dir~TLT\bgroup
4681     \tag_socket_use:nn{luamplib/figure/begin}{#2}
4682     \setbox\mplibscratchbox\hbox\bgroup
4683       \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
4684       \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4685       \egroup
4686       \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4687       \egroup
4688     \endgroup
4689 }

```

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

```

4690 \cs_new_nopar:Npn \mplibalttext #1
4691 {
4692   \tl_set:Nn \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4693 }
4694 \cs_new_nopar:Npn \mplibactualtext #1
4695 {
4696   \tl_set:Nn \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4697 }
4698 \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.	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.	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.
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.	4. 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: (a) 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 (b) 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, (c) 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 object 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.	No WARRANTY 12. 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. 13. 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.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	5. 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. 6. 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. 7. 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. 8. 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. 9. 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. 10. 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.	END OF TERMS AND CONDITIONS Appendix: How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. one line to give the program's name and a brief idea of what it does. Copyright (C) yyyy name of author This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA. Also add information on how to contact you by electronic and paper mail. If the program is interactive, make it output a short notice like this when it starts in an interactive mode: Gnomovision version 69, Copyright (C) yyyy name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details. The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program. You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names: Yoyodyne, Inc., hereby disclaims all copyright interest in the program "Gnomovision" (which makes passes at compilers) written by James Hacker. signature of Ty Coon, 1 April 1989 Ty Coon, President of Vice This General Public License does not permit incorporating your program into proprietary programs. If your program is a subcomponent library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.