

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/08/20 v2.42.8

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	115
3	The GNU GPL License v2	136

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 tested their proper functioning.

transparency (texdoc metafun § 8.2) Transparency is so simple that you can apply it to an object, with *plain* format as well as *metafun*, just by appending `withprescript "tr_transparency=<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 pallel 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 $\LaTeX$'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 L^AT_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";
```

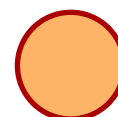
But the result may vary in its color model (gray/rgb/cmyk) according to the given $\TeX$ color. Therefore the example shown above would raise a METAPOST error: `cmykcolor col;` should have been declared. By contrast, `mplibrbgtexcolor` $\langle string \rangle$ always returns rgb-model expressions.

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) ;
```

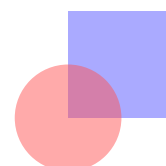


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

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
```

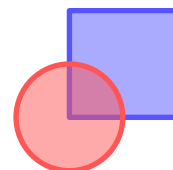


⁹Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color.

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.

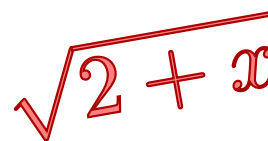
```
\mpfig
pickup pencircle scaled 2;
filldraw unitsquare scaled 40
  withcolor 1/3[blue,white]
  withmplibopacities (1, 1/2, 1)    % or ("normal", 1/2, 1)
;
filldraw fullcircle scaled 40
  withcolor 1/3[red,white]
  withmplibopacities (1, 1/2, 1)
;
\endmpfig
```



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 $\text{\TeX}$ 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.

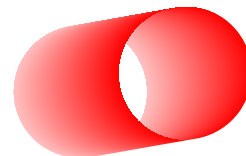
withshadingextend ($\langle boolean \rangle$, $\langle boolean \rangle$) Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

`\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).

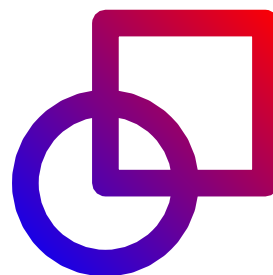
withshadingstroke $\langle \text{string} \rangle$ When the shading object is a $\langle \text{path} \rangle$, the string shall be "yes" or "no": if "yes", we get the path stroked and *then* shaded. It could be more efficient than issuing two sentences.

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

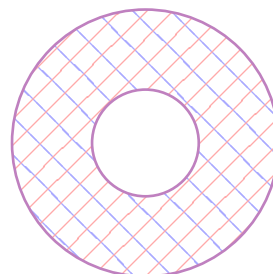
An example:

```

\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.

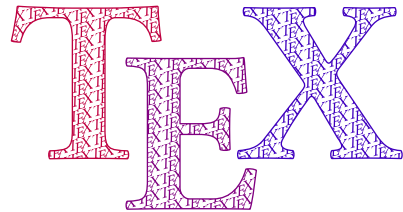
When you use special effect such as transparency in a pattern cell, *resources* option is

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}
```

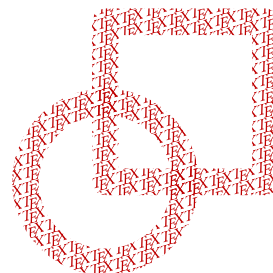


After v2.42.6, we provide an optional macro for both colored and uncolored patterns:

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:

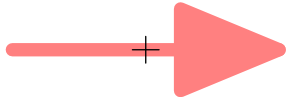
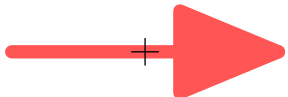
withgroupname $\langle string \rangle$ associates an XObject with the given name. When this command is not appended to the sentence with *asgroup* operator, the default name 'lastmplibgroup' will be used.

`\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.

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

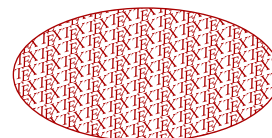
T_EX macros `\mplibgroup{<name>}` ... `\endmplibgroup` are described here in this subsection, as they are deeply related to the `asgroup` operator described just above at § 1.2.15. With these, users can define a transparency group or an ordinary *form XObject* from T_EX side. The syntax is similar to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

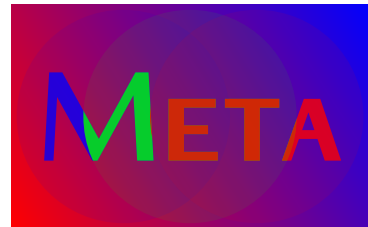
²⁰Color expressions in rgb or cmyk model are also allowed, in accordance with the option colorspace="/DeviceRGB"

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.

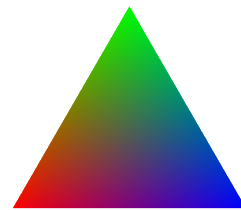
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates).

The third argument is the color for the new vertex.

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

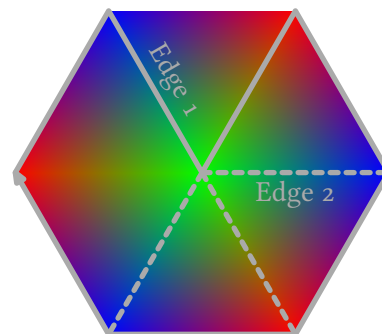
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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.

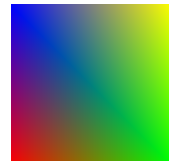
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

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.

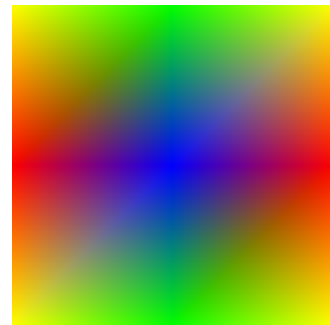
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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.

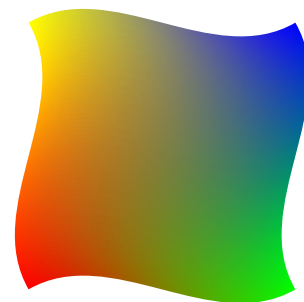
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

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.

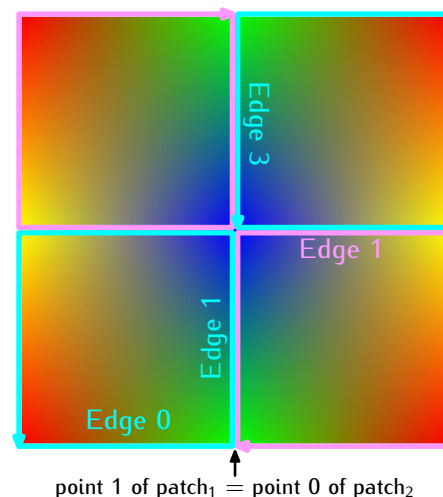
Remaining arguments are two color expressions for the new vertices.

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

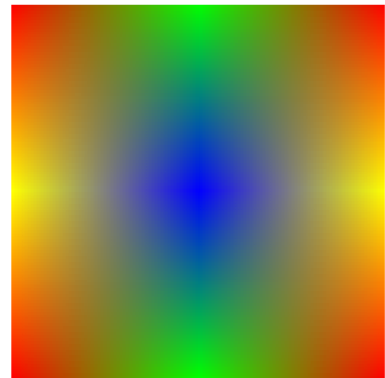
```

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.

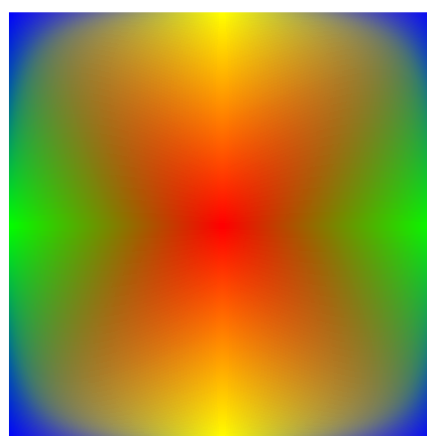
The only difference to withcoonspatchinit macro of Coons patch shading is the second argument inserted for specifying four inner control points. It shall be a path that has four points, or a string composed of eight numerics separated by spaces (x and y coordinates of each point). If the first argument 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, `luamplib.instances`, through which METAPOST variables are also easily accessible from Lua side, as documented in LuaTeX manual § 11.2.8.4 (texdoc luatex). The following example will print false, 3.0, MetaPost, and the knots and the cyclicity of the path `unitsquare`.

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

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

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

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

3.0

1.3.3 Running a METAPOST code

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

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

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

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

1.3.4 Registering a tiling pattern

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

```
luamplib.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> ($\langle\langle string \rangle\rangle$)	<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> ($\langle\langle string \rangle\rangle$)	<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    = "2.42.8",
5   date      = "2026/08/20",
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 .. "/_luam_plib_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

```

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

```

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

```

Function luamplib.finder

```

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

```

format.mp is much complicated, so specially treated.

```

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

```

```

181     return nf.size == 0 and file or newfile
182   end
183 end
184 if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
185 local fh = ioopen(file,"r")
186 if not fh then return file end
187 local data = fh:read("*all"); fh:close()

```

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

```

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

```

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

```

208 local mpkpse
209 do
210   local exe = 0
211   while arg[exe-1] do
212     exe = exe-1
213   end
214   mpkpse = kpse.new(arg[exe], "mpost")
215 end
216 local special_ftype = {
217   pfb = "type1 fonts",
218   enc = "enc files",
219 }
220 function luamplib.finder (name, mode, ftype)
221   if mode == "w" then
222     if name and name ~= "mpout.log" then
223       kpse.record_output_file(name) -- recorder
224     end

```

```

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

```

For the main function: process

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

```

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

```

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

```

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

```

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

```

260     local log = l or t or "no-term"
261     log = log:gsub("%(Please type a command or say `end'%)", ""):gsub("\n+", "\n")
262     if result.status > 0 then
263         local first = log:match("(.-\n! .-)\n! "
264         if first then
265             termorlog("term", first)
266             termorlog("log", log, "Warning")
267         else
268             warn(log)

```

```

269     end
270     if result.status > 1 then
271         err(e or "see above messages")
272     end
273     elseif prevlog then
274         log = prevlog..log

```

v2.6.1: now luamplib does not disregard show command, even when luamplib.showlog is false. Incidentally, it does not raise error nor prints an info, even if output has no figure.

```

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

```

lualibs-os.lua installs a randomseed. When this file is loaded, we should explicitly seed a unique integer to get random randomseed for each run.

```

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

```

Make use of make_text and run_script. And we provide numbersystem option since v2.4. See <https://github.com/lualatex/luamplib/issues/21>.

```

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

```

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

```

299 local preamble = tableconcat{
300     format(luamplib.preambles.preamble, replacesuffix(name,"mp")),
301     luamplib.preambles.mplibcode,
302     luamplib.legacyverbatim and luamplib.preambles.legacyverbatim or "",
303     luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
304 }
305 local result, log
306 if not mpx then
307     result = { status = 99, error = "out of memory"}
308 else

```

```

309     result = mpx:execute(preamble)
310 end
311 log = reporterror(result)
312 return mpx, result, log
313 end

```

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

```

314 local process_stack = 0
315 function process (data, instancename)
316     local currfmt
317     process_stack = process_stack + 1
318     if instancename and instancename ~= "" then
319         currfmt = instancename
320         has_instancename = true
321     else
322         currfmt = tableconcat{
323             currentformat,
324             luamplib.numbersystem or "scaled",
325             tostring(luamplib.texttextlabel),
326             tostring(luamplib.legacyverbatim),
327             tostring(process_stack), -- try to address #63
328         }
329         has_instancename = false
330     end
331     local mpx = mplibinstances[currfmt]
332     local standalone = not (has_instancename or luamplib.codeinherit)
333     if mpx and standalone then
334         mpx:finish()
335     end
336     local log = ""
337     if standalone or not mpx then
338         mpx, _, log = luamplibload(currentformat)
339         mplibinstances[currfmt] = mpx
340     end
341     local converted, result = false, {}
342     if mpx and data then
343         result = mpx:execute(data)
344         local log = reporterror(result, log)
345         if log then
346             if result.fig then
347                 converted = luamplib.convert(result)
348             end
349         end
350     else
351         err"Mem file unloadable. Maybe generated with a different version of mplib?"
352     end
353     process_stack = process_stack - 1
354     return converted, result
355 end
356 end

```

357

dvipdfmx is supported, though nobody seems to use it.

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

359

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

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

```
361 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.

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

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

```
364 end
```

For conversion of sp to bp.

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

366

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.

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

```
368 local process_tex_text
```

```
369 do
```

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

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

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

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

```
374     if str then
```

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

```
376       local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
377                     and "\\global" or ""
```

```
378       local tex_box_id
```

```
379       if global == "" then
```

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

```
381         texboxes.localid = tex_box_id
```

```
382       else
```

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

```
384         texboxes.globalid = boxid
```

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

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

```
387       end
```

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

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

```
390         run_tex_code(format("\\luamplibnotagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
391                               tex_box_id, global, tex_box_id, str))
```

```
392       else
```

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

```

394             tex_box_id, global, tex_box_id, str))
395     end
396     local box = texgetbox(tex_box_id)
397     local wd = box.width / factor
398     local ht = box.height / factor
399     local dp = box.depth / factor
400     return text_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
401 end
402 return ""
403 end
404 end
405

```

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.

```

406 if is_defined'color_select:n' then
407   run_tex_code{
408     "\newcatcodetable\luamplibcctabexplat",
409     "\begingroup",
410     "\catcode\@=11 ",
411     "\catcode\_ =11 ",
412     "\catcode\:=11 ",
413     "\savecatcodetable\luamplibcctabexplat",
414     "\endgroup",
415   }
416 end
417 local ccexplat = luatexbase.registernumber"luamplibcctabexplat"
418
419 local process_color, process_mplibcolor

```

A common function for color functions

```

420 local function is_xcolor (str)
421   for _,v in ipairs(str:explode"!") do
422     if not v:find("^%s*%d+%s*$") and is_defined("\color@".v) then -- priority to xcolor
423       return true
424     end
425   end
426   return false
427 end
428 local function colorsplit (res)
429   local t, tt = { }, res:explode()
430   local be = tt[1]:find"%" and 1 or 2
431   for i=be, #tt do
432     if not tonumber(tt[i]) then break end
433     t[#t+1] = tt[i]
434   end
435   return t
436 end
437 do
438   local colfmt = ccexplat and "l3color" or "xcolor"

```

```

439 local mplibcolorfmt = {
440   xcolor = [[{\setbox0\hbox{{\color%s\global\mplibmptoks\expandafter{\current@color}}}}]],
441   l3color = [[\color_export:nnN%s{raw}\l_tmpa_tl\mplibmptoks\expandafter{\l_tmpa_tl}]]
442 }
443 function process_color (str)
444   if str then
445     if not str:find("%b{") then
446       str = format("{%s}",str)
447     end
448     local myfmt = mplibcolorfmt[colfmt]
449     if colfmt == "l3color" and is_defined"color" then
450       if str:find("%b[") or is_xcolor(str:match"{{(.+)}}") then
451         myfmt = mplibcolorfmt.xcolor
452       end
453     end
454     run_tex_code(myfmt:format(str), ccexplat or catat11)
455     local t = texgettoks"mplibmptoks"
456     if not pdfmode then
457       if t:find"^hsb" then
458         local spec = t:gsub("^hsb ", ""):gsub(" ", ",")
459         run_tex_code{"\\convertcolorspec{hsb}{", spec, "} {rgb}\\mplibtmpa"}
460         t = format("rgb %s", get_macro"mplibtmpa":gsub(",", " "))
461       elseif not t:find"d" then -- named color
462         if not is_defined"extractcolorspecs" then err"needs xcolor package loaded" end
463         run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
464         t = format("%s %s", get_macro"mplibtmpa", get_macro"mplibtmpb":gsub(",", " "))
465       end
466       local name, value = t:match"^(%a+) (.+)"
467       if name and value then
468         local op = name == "rgb" and "rg" or name == "cmyk" and "k" or name == "gray" and "g"
469         if not op then err"unknown color model" end
470         t = ("%s %s %s %s"):format(value, op, value, op:upper())
471       end
472       elseif is_defined"ver@colorspace.sty" and t:find"/&" then
473         local a,b,c,d = t:match"^(.- cs) (.- CS) (.- scn?) (.- SCN?)$"
474         if a and b and c and d then
475           t = tableconcat({ a, c, b, d }, " ")
476         end
477       end
478       return format('1 withprescript "mpliboverridecolor=%s"', t)
479     end
480     return ""
481   end
482 function process_mplibcolor(str)
483   local res = process_color(str)
484   if res:find" cs " then return res end
485   res = colorsplit(res:match"mpliboverridecolor=(.+)")
486   return format("(%s)", tableconcat(res, ","))
487 end

```

```

488 end
489
    for \mpdim or mplibdimen
490 local function process_dimen (str)
491   if str then
492     str = str:gsub("{(.+)}", "%1")
493     run_tex_code(format([[\\mplibtmp toks\\expandafter{\\the\\dimexpr %s\\relax}]], str))
494     return format("begingroup %s endgroup", texgettoks"mplibtmp toks")
495   end
496   return ""
497 end
498

```

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

```

499 local function process_verbatimtex_text (str)
500   if str then
501     run_tex_code(str)
502   end
503   return ""
504 end
505

```

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.

```

506 local tex_code_pre_mplib = {}
507 luamplib.figid = 1
508 luamplib.in_the_fig = false
509 local function process_verbatimtex_prefig (str)
510   if str then
511     tex_code_pre_mplib[luamplib.figid] = str
512   end
513   return ""
514 end
515 local function process_verbatimtex_infig (str)
516   if str then
517     return format('special "postmplibverbtx=%s";', str)
518   end
519   return ""
520 end
521

```

For *metafun* format. see issue #79.

```

522 mp = mp or {}
523 local mp = mp
524 mp.mf_path_reset = mp.mf_path_reset or function() end
525 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
526 mp.report = mp.report or info

```

metafun 2021-03-09 changes crashes luamplib.

```

527 catcodes = catcodes or {}

```

```

528 local catcodes = catcodes
529 catcodes.numbers = catcodes.numbers or {}
530 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
531 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
532 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
533 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
534 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
535 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
536 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
537

```

Now `luamplib.runscript`

```

538 do
539   local runscript_funcs = {
540     luamplibtext    = process_tex_text,
541     luamplibcolor   = process_mplibcolor,
542     luamplibdimen   = process_dimen,
543     luamplibprefig  = process_verbatimtex_prefig,
544     luamplibinfig   = process_verbatimtex_infig,
545     luamplibverbtex = process_verbatimtex_text,
546   }

```

A function from `ConTeXt` general.

```

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

```

```

574   if type(f) == "function" then
575     local buffer = {}
576     function mp.print(...)
577       mpprint(buffer,...)
578     end
579     local res = {f()}
580     buffer = tableconcat(buffer)
581     if buffer and buffer ~= "" then
582       return buffer
583     end
584     buffer = {}
585     mpprint(buffer, tableunpack(res))
586     return tableconcat(buffer)
587   end
588   return ""
589 end
590 end
591

```

luamplib.maketext

```

592 luamplib.legacyverbatimtex = true
593 do

```

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

```

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

```

```

620     else
621         return process_tex_text(str, true) -- bool is for 'char13'
622     end
623 end
624 return ""
625 end
626 end
627
    luamplib's METAPOST color operators
628 local function get_spc_name_obj (spot)
629     if is_defined"spc@csall" then
630         for v in get_macro"spc@csall":gmatch"{(.-)}" do
631             local n, r = get_macro("spc@ir@".v):match"^(.-) (%d+ 0 R)"
632             if spot == n then
633                 return v, r
634             end
635         end
636     else
637         err"only l3color or colorspace is supported for spot color"
638     end
639 end
640 do
641     local function cmyk2rgb (t)
642         return {
643             1 - math.min(1, t[1]+t[4]),
644             1 - math.min(1, t[2]+t[4]),
645             1 - math.min(1, t[3]+t[4]),
646         }
647     end
648     luamplib.gettexcolor = function (str, rgb)
649         local res = process_color(str):match"mpliboverridecolor=(.+)
650         if res:find" cs " then
651             info("%s is a spot color. Forced to %s", str, rgb and "RGB" or "CMYK")
652             if not is_xcolor(str) then
653                 run_tex_code({
654                     "\\color_export:nnN{" ,
655                     str,
656                     "}" ,
657                     rgb and "space-sep-rgb" or "space-sep-cmyk",
658                     "\\mplib@tempa",
659                 },ccexplat)
660                 return get_macro"mplib@tempa":explode()
661             else
662                 local name, value = res:match"^(.-) cs (.-) sc"
663                 local t = get_macro("spc@ascmyk@".get_spc_name_obj(name)):explode"," -- mixed not work
664                 for i = 1, #t do
665                     t[i] = t[i] * value
666                 end
667                 return rgb and cmyk2rgb(t) or t

```

```

668     end
669 end
670 local t = colorsplit(res)
671 if #t == 3 or not rgb then return t end
672 if #t == 4 then return cmyk2rgb(t) end
673 return { t[1], t[1], t[1] }
674 end
675 end
676 luamplib.shadecolor = function (str)
677 local res = process_color(str):match'"mpliboverridecolor=(.+)"'
678 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

```

\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}

679   local name, value, obj
680   if not is_xcolor(str) then
681     run_tex_code({ "\color_export:nnN{" , str, "{backend}\mplib@tempa" }, ccexplat)
682     name, value = get_macro'mplib@tempa':match'{{(.-)}{(.-)}}'
683     local t = res:explode()
684     local refname = t[1]:sub(2,-1)
685     if pdfmode then
686       obj = format("%s 0 R", ltx.pdf.object_id(refname))
687     else
688       run_tex_code({ "\mplibtmp toks\expanded{{\pdf_object_ref:n{" , refname, "}}}" }, ccexplat)
689       obj = texgettoks"mplibtmp toks"
690     end
691   else -- colorspace: mixing is allowed only in preamble
692     name, value = res:match"^(.-) cs (.-) sc"
693     name, obj = get_spc_name_obj(name)
694   end
695   return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
696 end

```

```

697 return format('(%)s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),","), str)
698 end
699

```

luamplib.fillandstrokecolor

```

700 do
701 local function graphictextcolor (col, filldraw)
702   if col:find"^[%d%.:]+$" then
703     col = col:explode":"
704     for i=1,#col do
705       col[i] = format("%.3f", col[i])
706     end
707     local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
708     col[#col+1] = filldraw == "fill" and op or op:upper()
709     return tableconcat(col," ")
710   end
711   col = process_color(col):match'"mpliboverridecolor=(.+)"'
712   local t = col:explode()
713   local b = filldraw == "fill" and 1 or #t/2+1
714   local e = b == 1 and #t/2 or #t
715   return tableconcat(t," ", b, e)
716 end
717 function luamplib.fillandstrokecolor (fill, stroke)
718   fill = graphictextcolor(fill, "fill")
719   stroke = graphictextcolor(stroke, "stroke")
720   return format('withprescript "mpliboverridecolor=%s %s"', fill, stroke)
721 end
722 end
723

```

Remove trailing zeros for smaller PDF

```

724 local decimals = "%. %d+"
725 local function rmzeros(str) return str:gsub("%.?0+$", "") end
726

```

common function for mplibgraphictext and mpliboutlinetext

```

727 local function getrulemetric (box, curr, bp)
728   local running = -1073741824
729   local wd,ht,dp = curr.width, curr.height, curr.depth
730   wd = wd == running and box.width or wd
731   ht = ht == running and box.height or ht
732   dp = dp == running and box.depth or dp
733   if bp then
734     return wd/factor, ht/factor, dp/factor
735   end
736   return wd, ht, dp
737 end
738

```

luamplib's mplibgraphictext operator

```

739 do
740   if not math.round then
741     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
742   end
743   local emboldenfonts = { }
744   local function roundupwidth (f, fb)
745     local wd = math.round(f.size * fb / factor * 10)
746     if wd == 0 and fb ~= 0 then
747       wd = 1
748     end
749     emboldenfonts.width = wd
750     return wd
751   end
752   local function getemboldenwidth (curr, fakebold)
753     local width = emboldenfonts.width
754     if not width then
755       local f
756       local function getglyph(n)
757         while n do
758           if n.head then
759             getglyph(n.head)
760           elseif n.font and n.font > 0 then
761             f = n.font; break
762           end
763           n = node.getnext(n)
764         end
765       end
766       getglyph(curr)
767       width = roundupwidth(font.getcopy(f or font.current()), fakebold)
768     end
769     return width
770   end
771   local function getrulewhatsit (line, wd, ht, dp)
772     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
773     line = line == 0 and "" or ("%f w"):format(line)
774     local pl
775     local fmt = "q %s %f %f %f %f re B Q"
776     if pdfmode then
777       pl = node.new("whatsit", "pdf_literal")
778       pl.mode = 0
779     else
780       fmt = "pdf:content " .. fmt
781       pl = node.new("whatsit", "special")
782     end
783     pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
784     local ss = node.new"glue"
785     node.setglue(ss, 0, 65536, 65536, 2, 2)
786     pl.next = ss
787     return pl

```

```
788 end
```

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

```
789 local tag_update_attrs
790 if is_defined"ver@tagpdf.sty" then
791   tag_update_attrs = function (n, curr)
792     while n do
793       n.attr = curr.attr
794       if n.head then
795         tag_update_attrs(n.head, curr)
796       end
797       n = node.getnext(n)
798     end
799   end
800 else
801   tag_update_attrs = function() end
802 end
803 local function embolden (box, curr, fakebold)
804   local head = curr
805   while curr do
806     if curr.head then
807       curr.head = embolden(curr, curr.head, fakebold)
808     elseif curr.replace then
809       curr.replace = embolden(box, curr.replace, fakebold)
810     elseif curr.leader then
811       if curr.leader.head then
812         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
813       elseif curr.leader.id == node.id"rule" then
814         local glue = node.effective_glue(curr, box)
815         local line = getemboldenwidth(curr, fakebold)
816         local wd,ht,dp = getrulemetric(box, curr.leader)
817         if box.id == node.id"hlist" then
818           wd = glue
819         else
820           ht, dp = 0, glue
821         end
822         local pl = getrulewhatsit(line, wd, ht, dp)
823         local pack = box.id == node.id"hlist" and node.hpack or node.vpack
824         local list = pack(pl, glue, "exactly")
825         tag_update_attrs(list,curr)
826         head = node.insert_after(head, curr, list)
827         head, curr = node.remove(head, curr)
828       end
829     elseif curr.id == node.id"rule" and curr.subtype == 0 then
830       local line = getemboldenwidth(curr, fakebold)
831       local wd,ht,dp = getrulemetric(box, curr)
832       if box.id == node.id"vlist" then
833         ht, dp = 0, ht+dp
834       end
```

```

835     local pl = getrulewhatsit(line, wd, ht, dp)
836     local list
837     if box.id == node.id"hlist" then
838         list = node.hpack(pl, wd, "exactly")
839     else
840         list = node.vpack(pl, ht+dp, "exactly")
841     end
842     tag_update_attrs(list,curr)
843     head = node.insert_after(head, curr, list)
844     head, curr = node.remove(head, curr)
845 elseif curr.id == node.id"glyph" and curr.font > 0 then
846     local f = curr.font
847     local key = format("%s:%s",f,fakebold)
848     local i = emboldenfonts[key]
849     if not i then
850         local ft = font.getfont(f) or font.getcopy(f)
851         local width = roundupwidth(ft, fakebold)
852         if ft.format == "opentype" or ft.format == "truetype" then
853             local name = ft.name:gsub('',''):gsub(';','$','')
854             local t = name:gsub("^file:", ""):gsub("^name:", ""):gsub("^kpse:", ""):gsub("^my:", "")
855             name = format('%s%sembolden=%s;',name, t:find":" and ";" or ":", fakebold)
856             _, i = fonts.constructors.readanddefine(name,ft.size)
857         elseif pdfmode then
858             local ft = table.copy(ft)
859             ft.mode, ft.width = 2, width
860             i = font.define(ft)
861         else
862             goto skip_type1
863         end
864         emboldenfonts[key] = i
865     end
866     curr.font = i
867 end
868 ::skip_type1::
869 curr = node.getnext(curr)
870 end
871 return head
872 end
873 luamplib.graphicstext = function (text, fakebold, fc, dc)
874     local fmt = process_tex_text(text):sub(1,-2)
875     local id = tonumber(fmt:match"mplibtexboxid=(%d+)")
876     emboldenfonts.width = nil
877     local box = texgetbox(id)
878     box.head = embolden(box, box.head, fakebold)
879     local colors = luamplib.fillandstrokecolor(fc, dc)
880     return format('%s %s)', fmt, colors)
881 end
882 end
883

```

luamplib's mplibglyph operator

```

884 do
885   local function mperr (str)
886     return format("hide(errmessage %q)", str)
887   end
888   local function getangle (a,b,c)
889     local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
890     if r > 180 then
891       r = r - 360
892     elseif r < -180 then
893       r = r + 360
894     end
895     return r
896   end
897   local function turning (t)
898     local r, n = 0, #t
899     for i=1,2 do
900       tableinsert(t, t[i])
901     end
902     for i=1,n do
903       r = r + getangle(t[i], t[i+1], t[i+2])
904     end
905     return r/360
906   end
907   local function glyphimage(t, fmt)
908     local q, p, r, towarn = {},{}
909     local function closepath(dots)
910       tableinsert(p, format("%scycle", dots or "--"))
911       tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
912     end
913     for i,v in ipairs(t) do
914       local cmd = v[#v]
915       local nt = t[i+1]
916       local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
917       if cmd == "m" then
918         if final then towarn = true end
919         p = {format('(%s,%s)',v[1],v[2])}
920         r = {{x=v[1],y=v[2]}}
921       else
922         if cmd == "l" then
923           local pt = t[i-1]
924           if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
925             else
926               tableinsert(p, format('--(%s,%s)',v[1],v[2]))
927               tableinsert(r, {x=v[1],y=v[2]})
928             end
929           if final then closepath() end
930         elseif cmd == "c" then
931           tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))

```

```

932         if final and r[1].x == v[5] and r[1].y == v[6] then
933             closepath ".."
934         else
935             tableinsert(p, format('..(%s,%s)',v[5],v[6]))
936             tableinsert(r, {x=v[5],y=v[6]})
937             if final then closepath() end
938         end
939         elseif cmd == "path" or cmd == "move" then
940             else
941                 return mperr"unknown operator"
942             end
943         end
944     end
945     r = { }
946     if fmt == "opentype" then
947         for _,v in ipairs(q[1]) do
948             tableinsert(r, format('addto currentpicture contour %s;',v))
949         end
950         for _,v in ipairs(q[2]) do
951             tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
952         end
953     else
954         for _,v in ipairs(q[2]) do
955             tableinsert(r, format('addto currentpicture contour %s;',v))
956         end
957         for _,v in ipairs(q[1]) do
958             tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
959         end
960     end
961     return format('image(%s)', tableconcat(r)), towarn
962 end
963 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
964 function luamplib.glyph (f, c)
965     local filename, subfont, instance, kind, shapedata
966     local fid = tonumber(f) or font.id(f)
967     if fid > 0 then
968         local fontdata = font.getfont(fid) or font.getcopy(fid)
969         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
970         instance = fontdata.specification and fontdata.specification.instance
971         or fontdata.shared and fontdata.shared.features.axis
972         filename = filename and filename:gsub("^harfloaded:", "")
973     else
974         local name
975         f = f:match"^%s*(.+)%.s*$"
976         name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
977         if not name then
978             name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
979         end
980         if not name then

```

```

981     name, subfont = f:match"(.)%((%d+)%)$" -- Times.ttc(2)
982 end
983 name = name or f
984 subfont = (subfont or 0)+1
985 instance = instance and instance:lower()
986 for _,ftype in ipairs{"opentype", "truetype"} do
987     filename = kpse.find_file(name, ftype.." fonts")
988     if filename then
989         kind = ftype; break
990     end
991 end
992 end
993 if kind ~= "opentype" and kind ~= "truetype" then
994     f = fid and fid > 0 and tex.fontname(fid) or f
995     if kpse.find_file(f, "tfm") then
996         return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
997     else
998         filename = kpse.find_file(f, "type1 fonts")
999         if filename then
1000             kind = "type1" -- there's bug in processing cmr family
1001         else
1002             return mperr"font not found"
1003         end
1004     end
1005 end
1006 local time = lfsattributes(filename,"modification")

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

1007 local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1008 local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
1009 local newname = format("%s/%s.lua", cachedir or outputdir(), h)
1010 local newtime = lfsattributes(newname,"modification") or 0
1011 if time == newtime then
1012     shapedata = require(newname)
1013 end
1014 if not shapedata then
1015     if fonts then
1016         local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1017         shapedata = handler.readers.loadshapes(filename,subfont,instance)
1018     end
1019     if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1020     table.tofile(newname, shapedata, "return")
1021     lfstouch(newname, time, time)
1022 end
1023 local gid = tonumber(c)
1024 if not gid then
1025     local uni = utf8.codepoint(c)

```

```

1026     for i,v in pairs(shapedata.glyphs) do
1027         if c == v.name or uni == v.unicode then
1028             gid = i; break
1029         end
1030     end
1031 end
1032 if not gid then return mperr"cannot get GID (glyph id)" end
1033 local fac = 1000 / (shapedata.units or 1000)
1034 local t = shapedata.glyphs[gid]; t = t and t.segments
1035 if not t then return "image()" end
1036 for i,v in ipairs(t) do
1037     if type(v) == "table" then
1038         for ii,vv in ipairs(v) do
1039             if type(vv) == "number" then
1040                 t[i][ii] = format("%.0f", vv * fac)
1041             end
1042         end
1043     end
1044 end
1045 local result, towarn = glyphimage(t, shapedata.format or kind)
1046 if towarn then
1047     warn("mplibglyph %s not working properly. Use glyph instead", f)
1048 end
1049 return result
1050 end
1051 end
1052

```

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

```

1053 do
1054     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1055         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1056     local outline_horz, outline_vert
1057     function outline_vert (res, box, curr, xshift, yshift)
1058         local b2u = box.dir == "LTL"
1059         local dy = (b2u and -box.depth or box.height)/factor
1060         local ody = dy
1061         while curr do
1062             if curr.id == node.id"rule" then
1063                 local wd, ht, dp = getrulemetric(box, curr, true)
1064                 local hd = ht + dp
1065                 if hd ~= 0 then
1066                     dy = dy + (b2u and dp or -ht)
1067                     if wd ~= 0 and curr.subtype == 0 then
1068                         res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1069                     end
1070                     dy = dy + (b2u and ht or -dp)
1071                 end
1072             elseif curr.id == node.id"glue" then

```

```

1073     local vwidth = node.effective_glue(curr,box)/factor
1074     if curr.leader then
1075         local curr, kind = curr.leader, curr.subtype
1076         if curr.id == node.id"rule" then
1077             local wd = getrulemetric(box, curr, true)
1078             if wd ~= 0 then
1079                 local hd = vwidth
1080                 local dy = dy + (b2u and 0 or -hd)
1081                 if hd ~= 0 and curr.subtype == 0 then
1082                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1083                 end
1084             end
1085         elseif curr.head then
1086             local hd = (curr.height + curr.depth)/factor
1087             if hd <= vwidth then
1088                 local dy, n, iy = dy, 0, 0
1089                 if kind == 100 or kind == 103 then -- todo: gleaders
1090                     local ady = abs(ody - dy)
1091                     local ndy = math.ceil(ady / hd) * hd
1092                     local diff = ndy - ady
1093                     n = math.floor((vwidth-diff) / hd)
1094                     dy = dy + (b2u and diff or -diff)
1095                 else
1096                     n = math.floor(vwidth / hd)
1097                     if kind == 101 then
1098                         local side = vwidth % hd / 2
1099                         dy = dy + (b2u and side or -side)
1100                     elseif kind == 102 then
1101                         iy = vwidth % hd / (n+1)
1102                         dy = dy + (b2u and iy or -iy)
1103                     end
1104                 end
1105                 dy = dy + (b2u and curr.depth or -curr.height)/factor
1106                 hd = b2u and hd or -hd
1107                 iy = b2u and iy or -iy
1108                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1109                 for i=1,n do
1110                     res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1111                     dy = dy + hd + iy
1112                 end
1113             end
1114         end
1115     end
1116     dy = dy + (b2u and vwidth or -vwidth)
1117 elseif curr.id == node.id"kern" then
1118     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1119 elseif curr.id == node.id"vlist" then
1120     dy = dy + (b2u and curr.depth or -curr.height)/factor
1121     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)

```

```

1122     dy = dy + (b2u and curr.height or -curr.depth)/factor
1123 elseif curr.id == node.id"hlist" then
1124     dy = dy + (b2u and curr.depth or -curr.height)/factor
1125     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1126     dy = dy + (b2u and curr.height or -curr.depth)/factor
1127 end
1128 curr = node.getnext(curr)
1129 end
1130 return res
1131 end
1132 function outline_horz (res, box, curr, xshift, yshift, discwd)
1133     local r2l = box.dir == "TRT"
1134     local dx = r2l and (discwd or box.width/factor) or 0
1135     local dirs = { { dir = r2l, dx = dx } }
1136     while curr do
1137         if curr.id == node.id"dir" then
1138             local sign, dir = curr.dir:match"(.)(...)"
1139             local level, newdir = curr.level, r2l
1140             if sign == "+" then
1141                 newdir = dir == "TRT"
1142                 if r2l ~= newdir then
1143                     local n = node.getnext(curr)
1144                     while n do
1145                         if n.id == node.id"dir" and n.level+1 == level then break end
1146                         n = node.getnext(n)
1147                     end
1148                     n = n or node.tail(curr)
1149                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1150                 end
1151                 dirs[level] = { dir = r2l, dx = dx }
1152             else
1153                 local level = level + 1
1154                 newdir = dirs[level].dir
1155                 if r2l ~= newdir then
1156                     dx = dirs[level].dx
1157                 end
1158             end
1159             r2l = newdir
1160         elseif curr.char and curr.font and curr.font > 0 then
1161             local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1162             local gid = ft.characters[curr.char].index or curr.char
1163             local scale = ft.size / factor / 1000
1164             local slant = (ft.slant or 0)/1000
1165             local extend = (ft.extend or 1000)/1000
1166             local squeeze = (ft.squeeze or 1000)/1000
1167             local expand = 1 + (curr.expansion_factor or 0)/1000000
1168             local xscale, yscale = scale * extend * expand, scale * squeeze
1169             dx = dx - (r2l and curr.width/factor*expand or 0)
1170             local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor

```

```

1171 local xpos, ypos = dx + xshift + xoff, yshift + yoff
1172 local vertical = ""
1173 if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1174     if ft.shared.features.vertical then -- luatexko
1175         vertical = "rotated 90"
1176         local data = ft.characters[curr.char] or { }
1177         if ft.hb then
1178             local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1179             local charraise = (ft.luatexko_charraise or 0)/factor
1180             xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1181         else
1182             local cmds = data.commands or { {0,0}, {0,0} }
1183             local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1184             xpos, ypos = xpos + hoff, ypos + voff
1185         end
1186     elseif curr ~= box.head then -- luatexja
1187         vertical = "rotated 90"
1188         local en = ft.parameters.quad/factor/2
1189         xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1190     end
1191 end
1192 local image
1193 if ft.format == "opentype" or ft.format == "truetype" then
1194     image = luamplib.glyph(curr.font, gid)
1195 else
1196     local name, scale = ft.name, 1
1197     local vf = font.read_vf(name, ft.size)
1198     if vf and vf.characters[gid] then
1199         local cmds = vf.characters[gid].commands or { }
1200         for _,v in ipairs(cmds) do
1201             if v[1] == "char" then
1202                 gid = v[2]
1203             elseif v[1] == "font" and vf.fonts[v[2]] then
1204                 name = vf.fonts[v[2]].name
1205                 scale = vf.fonts[v[2]].size / ft.size
1206             end
1207         end
1208     end
1209     image = format("glyph %s of %q scaled %f", gid, name, scale)
1210 end
1211 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1212     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1213 dx = dx + (r2l and 0 or curr.width/factor*expand)
1214 elseif curr.replace then
1215     local width = node.dimensions(curr.replace)/factor
1216     dx = dx - (r2l and width or 0)
1217     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1218     dx = dx + (r2l and 0 or width)
1219 elseif curr.id == node.id"rule" then

```

```

1220     local wd, ht, dp = getrulemetric(box, curr, true)
1221     if wd ~= 0 then
1222         local hd = ht + dp
1223         dx = dx - (r2l and wd or 0)
1224         if hd ~= 0 and curr.subtype == 0 then
1225             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1226         end
1227         dx = dx + (r2l and 0 or wd)
1228     end
1229 elseif curr.id == node.id"glue" then
1230     local width = node.effective_glue(curr, box)/factor
1231     dx = dx - (r2l and width or 0)
1232     if curr.leader then
1233         local curr, kind = curr.leader, curr.subtype
1234         if curr.id == node.id"rule" then
1235             local wd, ht, dp = getrulemetric(box, curr, true)
1236             local hd = ht + dp
1237             if hd ~= 0 then
1238                 wd = width
1239                 if wd ~= 0 and curr.subtype == 0 then
1240                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1241                 end
1242             end
1243         elseif curr.head then
1244             local wd = curr.width/factor
1245             if wd <= width then
1246                 local dx = r2l and dx+width or dx
1247                 local n, ix = 0, 0
1248                 if kind == 100 or kind == 103 then -- todo: gleaders
1249                     local adx = abs(dx-dirs[1].dx)
1250                     local ndx = math.ceil(adx / wd) * wd
1251                     local diff = ndx - adx
1252                     n = math.floor((width-diff) / wd)
1253                     dx = dx + (r2l and -diff-wd or diff)
1254                 else
1255                     n = math.floor(width / wd)
1256                     if kind == 101 then
1257                         local side = width % wd / 2
1258                         dx = dx + (r2l and -side-wd or side)
1259                     elseif kind == 102 then
1260                         ix = width % wd / (n+1)
1261                         dx = dx + (r2l and -ix-wd or ix)
1262                     end
1263                 end
1264                 wd = r2l and -wd or wd
1265                 ix = r2l and -ix or ix
1266                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1267                 for i=1,n do
1268                     res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)

```

```

1269         dx = dx + wd + ix
1270     end
1271 end
1272 end
1273 end
1274 dx = dx + (r2l and 0 or width)
1275 elseif curr.id == node.id" kern" then
1276     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1277 elseif curr.id == node.id" math" then
1278     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1279 elseif curr.id == node.id" vlist" then
1280     dx = dx - (r2l and curr.width/factor or 0)
1281     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1282     dx = dx + (r2l and 0 or curr.width/factor)
1283 elseif curr.id == node.id" hlist" then
1284     dx = dx - (r2l and curr.width/factor or 0)
1285     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1286     dx = dx + (r2l and 0 or curr.width/factor)
1287 end
1288 curr = node.getnext(curr)
1289 end
1290 return res
1291 end
1292 function luamplib.outlinetext (text)
1293     local fmt = process_tex_text(text)
1294     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1295     local box = texgetbox(id)
1296     local res = outline_horz({ }, box, box.head, 0, 0)
1297     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1298     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1299 end
1300 end
1301

```

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

```

1302 function luamplib.getunicodegraphemes (s)
1303     local t = { }
1304     local graphemes = require'lua-uni-graphemes'
1305     for _, _, c in graphemes.graphemes(s) do
1306         table.insert(t, c)
1307     end
1308     return t
1309 end
1310 function luamplib.unicodesubstring (s,b,e,grph)
1311     local tt, t, step = { }
1312     if grph then
1313         t = luamplib.getunicodegraphemes(s)
1314     else
1315         t = { }

```

```

1316   for _, c in utf8.codes(s) do
1317       table.insert(t, utf8.char(c))
1318   end
1319 end
1320 if b <= e then
1321     b, step = b+1, 1
1322 else
1323     e, step = e+1, -1
1324 end
1325 for i = b, e, step do
1326     table.insert(tt, t[i])
1327 end
1328 s = table.concat(tt):gsub("'", "'&ditto'")
1329 return string.format("%s", s)
1330 end
1331

```

METAPOST preambles

```

1332 luamplib.preambles = {
1333   preamble = [[
1334 boolean mplib ; mplib := true ;
1335 let dump = endinput ;
1336 let normalfontsize = fontsize;
1337 input %s ;
1338 ]],
1339   mplibcode = [[
1340 texscriptmode := 2;
1341 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1342 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1343 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1344 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1345 if known context_mlib:
1346   defaultfont := "cmtt10";
1347   let infont = normalinfont;
1348   let fontsize = normalfontsize;
1349   vardef thelabel@#(expr p,z) =
1350     if string p :
1351       thelabel@#(p infont defaultfont scaled defaultscale,z)
1352     else :
1353       p shifted (z + labeloffset*mfun_laboff@# -
1354         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1355         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1356     fi
1357   enddef;
1358 else:
1359   vardef texttext@# primary t = rawtexttext (t) enddef;
1360   def message expr t =
1361     if string t: runscript("mp.report[="&t&"]=]") else: errmessage "Not a string" fi
1362   enddef;

```

```

1363 def withtransparency (expr a, t) =
1364   withprescript "tr_alternative=" & if numeric a: decimal fi a
1365   withprescript "tr_transparency=" & decimal t
1366 enddef;
1367 vardef ddecimal primary p =
1368   decimal xpart p & " " & decimal ypart p
1369 enddef;
1370 vardef boundingbox primary p =
1371   if (path p) or (picture p) :
1372     llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1373   else :
1374     origin
1375   fi -- cycle
1376 enddef;
1377 fi
1378 def resolvedcolor(expr s) =
1379   runscript("return luamplib.shadecolor('& s &''')")
1380 enddef;
1381 def colordecimals primary c =
1382   if cmykcolor c:
1383     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1384     decimal yellowpart c & ":" & decimal blackpart c
1385   elseif rgbcolor c:
1386     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1387   elseif string c:
1388     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1389   else:
1390     decimal c
1391   fi
1392 enddef;
1393 def externalfigure primary filename =
1394   draw rawtexttext("\includegraphics{'& filename &}")
1395 enddef;
1396 def TEX = texttext enddef;
1397 def mplibtexcolor primary c =
1398   runscript("return luamplib.gettexcolor('& c &''')")
1399 enddef;
1400 def mplibrgbtexcolor primary c =
1401   runscript("return luamplib.gettexcolor('& c &''', 'rgb')")
1402 enddef;
1403 def mplibgraphicstext primary t =
1404   begingroup;
1405   mplibgraphicstext_ (t)
1406 enddef;
1407 def mplibgraphicstext_ (expr t) text rest =
1408   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1409   fb, fc, dc, graphicstextpic, alsoordoublepath;
1410   picture graphicstextpic; graphicstextpic := nullpicture;
1411   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";

```

```

1412 let scale = scaled;
1413 def fakebold primary c = hide(fb:=c;) enddef;
1414 def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1415 def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1416 let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1417 def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1418 addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1419 def fakebold primary c = enddef;
1420 let fillcolor = fakebold; let drawcolor = fakebold;
1421 let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1422 image(draw runscript("return luamplib.graphictext([==["&t&"]==], "
1423   & decimal fb & ", ""& fc & ""', ""& dc & ""')") rest;))
1424 endgroup;
1425 enddef;
1426 def mplibglyph expr c of f =
1427   runscript (
1428     "return luamplib.glyph('"
1429     & if numeric f: decimal fi f
1430     & "'',"
1431     & if numeric c: decimal fi c
1432     & "')"
1433   )
1434 enddef;
1435 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1436 def mplibdrawglyph expr g =
1437   luamplib_tmp_num_ := 0;
1438   for item within g:
1439     fill pathpart item
1440     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1441   endfor
1442 enddef;
1443 let mplibfillglyph = mplibdrawglyph;
1444 def mplibstrokeglyph expr g =
1445   luamplib_tmp_num_ := 0;
1446   for item within g:
1447     draw pathpart item
1448     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1449   endfor
1450 enddef;
1451 def mplibfillandstrokeglyph expr g =
1452   luamplib_tmp_num_ := 0;
1453   for item within g:
1454     draw pathpart item withpostscript
1455     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1456   endfor
1457 enddef;
1458 def withmplibcolors (expr f, s) =
1459   runscript("return luamplib.fillandstrokecolor('" &
1460     if not string f: colordecimals fi f & "''," &

```

```

1461   if not string s: colordecimals fi s & "'')")
1462 enddef;
1463 def withmplibopacities (expr a, f, s) =
1464   withprescript "tr_alternative=" & if numeric a: decimal fi a
1465   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1466 enddef;
1467 def mplib_do_outline_text_set_b (text f) (text d) text r =
1468   def mplib_do_outline_options_f = f enddef;
1469   def mplib_do_outline_options_d = d enddef;
1470   def mplib_do_outline_options_r = r enddef;
1471 enddef;
1472 def mplib_do_outline_text_set_f (text f) text r =
1473   def mplib_do_outline_options_f = f enddef;
1474   def mplib_do_outline_options_r = r enddef;
1475 enddef;
1476 def mplib_do_outline_text_set_u (text f) text r =
1477   def mplib_do_outline_options_f = f enddef;
1478 enddef;
1479 def mplib_do_outline_text_set_d (text d) text r =
1480   def mplib_do_outline_options_d = d enddef;
1481   def mplib_do_outline_options_r = r enddef;
1482 enddef;
1483 def mplib_do_outline_text_set_r (text d) (text f) text r =
1484   def mplib_do_outline_options_d = d enddef;
1485   def mplib_do_outline_options_f = f enddef;
1486   def mplib_do_outline_options_r = r enddef;
1487 enddef;
1488 def mplib_do_outline_text_set_n text r =
1489   def mplib_do_outline_options_r = r enddef;
1490 enddef;
1491 def mplib_do_outline_text_set_p = enddef;
1492 def mplib_fill_outline_text =
1493   for n=1 upto mpliboutlinenum:
1494     i:=0;
1495     for item within mpliboutlinepic[n]:
1496       i:=i+1;
1497       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1498       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostsript "collect"; fi
1499     endfor
1500   endfor
1501 enddef;
1502 def mplib_draw_outline_text =
1503   for n=1 upto mpliboutlinenum:
1504     for item within mpliboutlinepic[n]:
1505       draw pathpart item mplib_do_outline_options_d;
1506     endfor
1507   endfor
1508 enddef;
1509 def mplib_filldraw_outline_text =

```

```

1510 for n=1 upto mpliboutlinenum:
1511   i:=0;
1512   for item within mpliboutlinepic[n]:
1513     i:=i+1;
1514     if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1515       fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1516     else:
1517       draw pathpart item mplib_do_outline_options_f withpostscript "both";
1518     fi
1519   endfor
1520 endfor
1521 enddef;
1522 vardef mpliboutlinetext@# (expr t) text rest =
1523   save kind; string kind; kind := str @#;
1524   save i; numeric i;
1525   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1526   def mplib_do_outline_options_d = enddef;
1527   def mplib_do_outline_options_f = enddef;
1528   def mplib_do_outline_options_r = enddef;
1529   runscript("return luamplib.outlinetext[==["&t&"]==]");
1530   image ( addto currentpicture also image (
1531     if kind = "f":
1532       mplib_do_outline_text_set_f rest;
1533       mplib_fill_outline_text;
1534     elseif kind = "d":
1535       mplib_do_outline_text_set_d rest;
1536       mplib_draw_outline_text;
1537     elseif kind = "b":
1538       mplib_do_outline_text_set_b rest;
1539       mplib_fill_outline_text;
1540       mplib_draw_outline_text;
1541     elseif kind = "u":
1542       mplib_do_outline_text_set_u rest;
1543       mplib_filldraw_outline_text;
1544     elseif kind = "r":
1545       mplib_do_outline_text_set_r rest;
1546       mplib_draw_outline_text;
1547       mplib_fill_outline_text;
1548     elseif kind = "p":
1549       mplib_do_outline_text_set_p;
1550       mplib_draw_outline_text;
1551     else:
1552       mplib_do_outline_text_set_n rest;
1553       mplib_fill_outline_text;
1554     fi;
1555   ) mplib_do_outline_options_r; )
1556 enddef ;
1557 def withmppattern primary p =
1558   withprescript "mplibpattern=" & if numeric p: decimal fi p

```

```

1559 enddef;
1560 def withpatternstroke expr a =
1561   withprescript "mplibpatternstroke=" & a
1562 enddef;
1563 primarydef t withpattern p =
1564   image(
1565     if cycle t:
1566       fill
1567     else:
1568       draw
1569     fi
1570     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1571 enddef;
1572 vardef mplibtransformmatrix (text e) =
1573   save t; transform t;
1574   t = identity e;
1575   runscript("luamplib.transformmatrix = {"
1576     & decimal xpart t & ","
1577     & decimal ypart t & ","
1578     & decimal xpart t & ","
1579     & decimal ypart t & ","
1580     & decimal xpart t & ","
1581     & decimal ypart t & ","
1582     & "}");
1583 enddef;
1584 primarydef p withmaskinggroup s =
1585   if picture p:
1586     image(
1587       draw p;
1588       draw center p withprescript "mplibfadestate=stop";
1589     )
1590   else:
1591     p withprescript "mplibfadestate=stop"
1592   fi
1593   withprescript "mplibfadetype=masking"
1594   withprescript "mplibmaskname=" & s
1595 enddef;
1596 def withmaskingbgcolor expr c =
1597   withprescript "mplibmaskingbgcolor=" & colordecimals c
1598 enddef;
1599 primarydef p withfademethod s =
1600   if picture p:
1601     image(
1602       draw p;
1603       draw center p withprescript "mplibfadestate=stop";
1604     )
1605   else:
1606     p withprescript "mplibfadestate=stop"
1607   fi

```

```

1608   withprescript "mplibfadetype=" & s
1609   hide(mplib_shade_step := 1;)
1610   withprescript "sh_color_a=1"
1611   withprescript "sh_color_b=0"
1612   withprescript "mplibfadebbox=" &
1613       decimal (xpart llcorner p -1/4) & ":" &
1614       decimal (ypart llcorner p -1/4) & ":" &
1615       decimal (xpart urcorner p +1/4) & ":" &
1616       decimal (ypart urcorner p +1/4)
1617 enddef;
1618 def withfadevector (expr a,b) =
1619   withprescript "mplibfadevector=" &
1620       decimal xpart a & ":" &
1621       decimal ypart a & ":" &
1622       decimal xpart b & ":" &
1623       decimal ypart b
1624 enddef;
1625 let withfadecenter = withfadevector;
1626 def withfaderadius (expr a,b) =
1627   withprescript "mplibfaderadius=" &
1628       decimal a & ":" &
1629       decimal b
1630 enddef;
1631 def withfadebbox (expr a,b) =
1632   withprescript "mplibfadebbox=" &
1633       decimal xpart a & ":" &
1634       decimal ypart a & ":" &
1635       decimal xpart b & ":" &
1636       decimal ypart b
1637 enddef;
1638 primarydef p asgroup s =
1639   image(
1640     draw center p
1641       withprescript "mplibgroupbbox=" &
1642           decimal (xpart llcorner p -1/4) & ":" &
1643           decimal (ypart llcorner p -1/4) & ":" &
1644           decimal (xpart urcorner p +1/4) & ":" &
1645           decimal (ypart urcorner p +1/4)
1646       withprescript "gr_state=start"
1647       withprescript "gr_type=" & s;
1648     draw p withprescript "sh_in_xobj=yes";
1649     draw center p withprescript "gr_state=stop";
1650   )
1651 enddef;
1652 def withgroupbbox (expr a,b) =
1653   withprescript "mplibgroupbbox=" &
1654       decimal xpart a & ":" &
1655       decimal ypart a & ":" &
1656       decimal xpart b & ":" &

```

```

1657     decimal ypart b
1658 enddef;
1659 def withgroupname expr s =
1660   withprescript "mplibgroupname=" & s
1661 enddef;
1662 def usemplibgroup primary s =
1663   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s &"\endcsname}")
1664   shifted runscript("return luamplib.trgroupshifts['' & s & ''"]")
1665 enddef;
1666 path    mplib_shade_path ;
1667 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1668 numeric mplib_shade_fx, mplib_shade_fy ;
1669 numeric mplib_shade_lx, mplib_shade_ly ;
1670 numeric mplib_shade_nx, mplib_shade_ny ;
1671 numeric mplib_shade_dx, mplib_shade_dy ;
1672 numeric mplib_shade_tx, mplib_shade_ty ;
1673 primarydef p withshadingmethod m =
1674   p
1675   if picture p :
1676     withprescript "sh_operand_type=picture"
1677     if textual p or (length p > 1):
1678       withprescript "sh_transform=no"
1679       mplib_with_shade_method (boundingbox p, m)
1680     else:
1681       withprescript "sh_transform=yes"
1682       mplib_with_shade_method (pathpart p, m)
1683     fi
1684   else :
1685     withprescript "sh_transform=yes"
1686     mplib_with_shade_method (p, m)
1687   fi
1688 enddef;
1689 def mplib_with_shade_method (expr p, m) =
1690   hide(mplib_with_shade_method_analyze(p))
1691   withprescript "sh_domain=0 1"
1692   withprescript "sh_color=into"
1693   withprescript "sh_color_a=" & colordecimals white
1694   withprescript "sh_color_b=" & colordecimals black
1695   withprescript "sh_first=" & ddecimal point 0 of p
1696   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1697   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1698   if m = "linear" :
1699     withprescript "sh_type=linear"
1700     withprescript "sh_factor=1"
1701     withprescript "sh_center_a=" & ddecimal llcorner p
1702     withprescript "sh_center_b=" & ddecimal urcorner p
1703   elseif m = "coons":
1704     withprescript "sh_type=coons"
1705     withprescript "sh_transform=no"

```

```

1706 elseif m = "triangle":
1707     withprescript "sh_type=triangle"
1708     withprescript "sh_transform=no"
1709 elseif m = "lattice":
1710     withprescript "sh_type=lattice"
1711     withprescript "sh_transform=no"
1712 elseif m = "tensor":
1713     withprescript "sh_type=tensor"
1714     withprescript "sh_transform=no"
1715     withprescript "sh_tensor_path=" &
1716         ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1717         ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1718         ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1719         ddecimal 1/4[point 3 of p, point 1 of p]
1720 else :
1721     withprescript "sh_type=circular"
1722     withprescript "sh_factor=1.2"
1723     withprescript "sh_center_a=" & ddecimal center p
1724     withprescript "sh_center_b=" & ddecimal center p
1725     withprescript "sh_radius_a=" & decimal 0
1726     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1727 fi
1728 enddef;
1729 def withlatticeverticesperrow expr n =
1730     withprescript "sh_lattice_perrow=" & decimal n
1731 enddef;
1732 def withlatticeverticesdata (text t) =
1733     hide(luamplib_tmp_num_ := 0;)
1734     withprescript "sh_lattice_data=" &
1735     for item = t:
1736         if odd(incr luamplib_tmp_num_):
1737             if pair item: ddecimal fi item & " " &
1738             fi
1739         endfor ""
1740     hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1741     for item = t:
1742         if not odd(incr luamplib_tmp_num_):
1743             withshadingstep( withshadingcolors(item,item) )
1744         fi
1745     endfor
1746 enddef;
1747 def withtrianglepatchinit (expr p, a, b, c) =
1748     if string p:
1749         withtrianglepatchnext(0, p , a)
1750         withtrianglepatchnext(0, "", b)
1751         withtrianglepatchnext(0, "", c)
1752     else:
1753         withtrianglepatchnext(0, point 0 of p, a)
1754         withtrianglepatchnext(0, point 1 of p, b)

```

```

1755   withtrianglepatchnext(0, point 2 of p, c)
1756   fi
1757 enddef;
1758 def withtrianglepatchnext (expr f, p, a) =
1759   withshadingstep(
1760     withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1761     withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1762       if string p: p else: ddecimal p fi
1763     withshadingcolors (a, a)
1764   )
1765 enddef;
1766 def withcoonspatchinit (expr p, a, b, c, d) =
1767   withshadingstep(
1768     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1769     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1770       if string p: p
1771       else:
1772         ddecimal point      0 of p & " " &
1773         ddecimal postcontrol 0 of p & " " &
1774         ddecimal precontrol  1 of p & " " &
1775         ddecimal point      1 of p & " " &
1776         ddecimal postcontrol 1 of p & " " &
1777         ddecimal precontrol  2 of p & " " &
1778         ddecimal point      2 of p & " " &
1779         ddecimal postcontrol 2 of p & " " &
1780         ddecimal precontrol  3 of p & " " &
1781         ddecimal point      3 of p & " " &
1782         ddecimal postcontrol 3 of p & " " &
1783         ddecimal precontrol  0 of p
1784       fi
1785     withshadingcolors (a, b)
1786   )
1787   withshadingstep ( withshadingcolors (c, d) )
1788 enddef;
1789 def withtensorpatchinit (expr p, q, a, b, c, d) =
1790   withshadingstep(
1791     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1792     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1793       if string p: p & " " & q
1794       else:
1795         ddecimal point      0 of p & " " &
1796         ddecimal postcontrol 0 of p & " " &
1797         ddecimal precontrol  1 of p & " " &
1798         ddecimal point      1 of p & " " &
1799         ddecimal postcontrol 1 of p & " " &
1800         ddecimal precontrol  2 of p & " " &
1801         ddecimal point      2 of p & " " &
1802         ddecimal postcontrol 2 of p & " " &
1803         ddecimal precontrol  3 of p & " " &

```

```

1804         ddecimal point      3 of p & " " &
1805         ddecimal postcontrol 3 of p & " " &
1806         ddecimal precontrol  0 of p & " " &
1807         ddecimal point      0 of q & " " &
1808         ddecimal point      1 of q & " " &
1809         ddecimal point      2 of q & " " &
1810         ddecimal point      3 of q
1811     fi
1812     withshadingcolors (a, b)
1813 )
1814 withshadingstep ( withshadingcolors (c, d) )
1815 enddef;
1816 def withcoonspatchnext (expr f, p, a, b) =
1817     withshadingstep(
1818         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1819         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1820         if string p: p
1821         else:
1822             ddecimal postcontrol 1 of p & " " &
1823             ddecimal precontrol  2 of p & " " &
1824             ddecimal point      2 of p & " " &
1825             ddecimal postcontrol 2 of p & " " &
1826             ddecimal precontrol  3 of p & " " &
1827             ddecimal point      3 of p & " " &
1828             ddecimal postcontrol 3 of p & " " &
1829             ddecimal precontrol  0 of p
1830         fi
1831         withshadingcolors (a, b)
1832     )
1833 enddef;
1834 def withtensorpatchnext (expr f, p, q, a, b) =
1835     withshadingstep(
1836         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1837         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1838         if string p: p & " " & q
1839         else:
1840             ddecimal postcontrol 1 of p & " " &
1841             ddecimal precontrol  2 of p & " " &
1842             ddecimal point      2 of p & " " &
1843             ddecimal postcontrol 2 of p & " " &
1844             ddecimal precontrol  3 of p & " " &
1845             ddecimal point      3 of p & " " &
1846             ddecimal postcontrol 3 of p & " " &
1847             ddecimal precontrol  0 of p & " " &
1848             ddecimal point      0 of q & " " &
1849             ddecimal point      1 of q & " " &
1850             ddecimal point      2 of q & " " &
1851             ddecimal point      3 of q
1852         fi

```

```

1853   withshadingcolors (a, b)
1854 )
1855 enddef;
1856 def mplib_with_shade_method_analyze(expr p) =
1857   mplib_shade_path := p ;
1858   mplib_shade_step := 1 ;
1859   mplib_shade_fx   := xpart point 0 of p ;
1860   mplib_shade_fy   := ypart point 0 of p ;
1861   mplib_shade_lx   := mplib_shade_fx ;
1862   mplib_shade_ly   := mplib_shade_fy ;
1863   mplib_shade_nx   := 0 ;
1864   mplib_shade_ny   := 0 ;
1865   mplib_shade_dx   := abs(mplib_shade_fx - mplib_shade_lx) ;
1866   mplib_shade_dy   := abs(mplib_shade_fy - mplib_shade_ly) ;
1867   for i=1 upto length(p) :
1868     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1869     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1870     if mplib_shade_tx > mplib_shade_dx :
1871       mplib_shade_nx := i + 1 ;
1872       mplib_shade_lx := xpart point i of p ;
1873       mplib_shade_dx := mplib_shade_tx ;
1874     fi ;
1875     if mplib_shade_ty > mplib_shade_dy :
1876       mplib_shade_ny := i + 1 ;
1877       mplib_shade_ly := ypart point i of p ;
1878       mplib_shade_dy := mplib_shade_ty ;
1879     fi ;
1880   endfor ;
1881 enddef;
1882 vardef mplib_max_radius(expr p) =
1883   max (
1884     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1885     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1886     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1887     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1888   )
1889 enddef;
1890 def withshadingstep (text t) =
1891   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1892   withprescript "sh_step=" & decimal mplib_shade_step
1893   t
1894 enddef;
1895 let withfadestep = withshadingstep;
1896 def withshadingradius expr a =
1897   withprescript "sh_radius_a=" & decimal (xpart a)
1898   withprescript "sh_radius_b=" & decimal (ypart a)
1899 enddef;
1900 def withshadingorigin expr a =
1901   withprescript "sh_center_a=" & decimal a

```

```

1902 withprescript "sh_center_b=" & ddecimal a
1903 enddef;
1904 def withshadingvector expr a =
1905   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1906   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1907 enddef;
1908 def withshadingdirection expr a =
1909   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1910   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1911 enddef;
1912 def withshadingtransform expr a =
1913   withprescript "sh_transform=" & a
1914 enddef;
1915 def withshadingcenter expr a =
1916   withprescript "sh_center_a=" & ddecimal (
1917     center mplib_shade_path shifted (
1918       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1919       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1920     )
1921   )
1922 enddef;
1923 def withshadingcenters (expr a, b) =
1924   withprescript "sh_center_a=" & ddecimal a
1925   withprescript "sh_center_b=" & ddecimal b
1926   withshadingtransform "no"
1927   withshadingfactor 1
1928 enddef;
1929 let withshadingpoints = withshadingcenters;
1930 def withshadingextend (expr a, b) =
1931   withprescript "sh_extend=" &
1932   if a: "true" else: "false" fi & " " &
1933   if b: "true" else: "false" fi
1934 enddef;
1935 let withfadeextend = withshadingextend;
1936 def withshadingdomain expr d =
1937   withprescript "sh_domain=" & ddecimal d
1938 enddef;
1939 def withshadingfactor expr f =
1940   withprescript "sh_factor=" & decimal f
1941 enddef;
1942 def withshadingfraction expr a =
1943   if mplib_shade_step > 0 :
1944     withprescript "sh_fraction=" & decimal mplib_shade_step & "=" & decimal a
1945   fi
1946 enddef;
1947 let withfadefraction = withshadingfraction;
1948 def withshadingcolors (expr a, b) =
1949   if mplib_shade_step > 0 :
1950     withprescript "sh_color=into"

```

```

1951   withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1952   withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1953   else :
1954     withprescript "sh_color=into"
1955     withprescript "sh_color_a=" & colordecimals a
1956     withprescript "sh_color_b=" & colordecimals b
1957   fi
1958 enddef;
1959 let withfadeopacity = withshadingcolors;
1960 def withshadingstroke expr a =
1961   withprescript "sh_stroking=" & a
1962 enddef;
1963 def withshadingmatrix expr s =
1964   withprescript "sh_matrix=" & s
1965 enddef;
1966 let withfadematrix = withshadingmatrix;
1967 def mpliblength primary t =
1968   runscript("return utf8.len[===[" & t & "]==]")
1969 enddef;
1970 def mplibsubstring expr p of t =
1971   runscript("return luamplib.unicodesubstring([===[" & t & "]==], "
1972     & decimal xpart p & ", "
1973     & decimal ypart p & ")")
1974 enddef;
1975 def mplibuclength primary t =
1976   runscript("return #luamplib.getunicodegraphemes[===[" & t & "]==]")
1977 enddef;
1978 def mplibucsubstring expr p of t =
1979   runscript("return luamplib.unicodesubstring([===[" & t & "]==], "
1980     & decimal xpart p & ", "
1981     & decimal ypart p & ", true)")
1982 enddef;
1983 ]],
1984 legacyverbatimtex = [[
1985 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1986 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1987 let VerbatimTeX = specialVerbatimTeX;
1988 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1989   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1990 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1991   "runscript(" &ditto&
1992   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1993   "luamplib.in_the_fig=false" &ditto& ");";
1994 ]],
1995 textxlabel = [[
1996 let luampliboriginalinfont = infont;
1997 primarydef s infont f =
1998   if (s < char 32)
1999     or (s = char 35) % #

```

```

2000    or (s = char 36) % $
2001    or (s = char 37) % %
2002    or (s = char 38) % &
2003    or (s = char 92) % \
2004    or (s = char 94) % ^
2005    or (s = char 95) % _
2006    or (s = char 123) % {
2007    or (s = char 125) % }
2008    or (s = char 126) % ~
2009    or (s = char 127) :
2010    s luampliboriginalinfont f
2011  else :
2012    rawtexttext(s)
2013  fi
2014 enddef;
2015 def fontsize expr f =
2016   begingroup
2017   save size; numeric size;
2018   size := mplibdimen("1em");
2019   if size = 0: 10pt else: size fi
2020 endgroup
2021 enddef;
2022 ]],
2023 }
2024

```

process_mplibcode

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

```

2025 luamplib.verbatiminput = false
2026 luamplib.everymplib      = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
2027 luamplib.everyendmplib   = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
2028 function luamplib.process_mplibcode (data, instancename)
2029   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2030 if luamplib.legacyverbatim then
2031   luamplib.figid, tex_code_pre_mplib = 1, {}
2032 end
2033 local everymplib      = luamplib.everymplib[instancename]
2034 local everyendmplib   = luamplib.everyendmplib[instancename]
2035 data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
2036 :gsub("\r","\n")

```

These five lines are needed for mplibverbatim mode.

```

2037 if luamplib.verbatiminput then
2038   data = data:gsub("\mpcolor%+{.-%b{}}", "mplibcolor(\"%1\")")
2039   :gsub("\mpdim%+{(%b{})}", "mplibdimen(\"%1\")")
2040   :gsub("\mpdim%+{(\%a+)}", "mplibdimen(\"%1\")")
2041   :gsub(btex_etex, "btex %1 etex ")
2042   :gsub(verbatimetex_etex, "verbatimetex %1 etex;")

```

```
2043 else
```

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

```
2044 local t = { } -- to store btex, verbatimtex, string
2045 data = data:gsub(btex_etex, function(str)
2046     t[#t+1] = str
2047     return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
2048 end)
2049 :gsub(verbatimtex_etex, function(str)
2050     t[#t+1] = str
2051     return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
2052 end)
2053 :gsub('"(.)"', function(str)
2054     t[#t+1] = str
2055     return format('"\\unexpanded{!l!u!a!%s!m!p!l!}"', #t)
2056 end)
2057 :gsub("\\%", "\\0PerCent\\0")
2058 :gsub("%%. -\\n", "\\n")
2059 :gsub("%zPerCent%z", "\\%")
2060 run_tex_code(format("\\mplibtmptoks\\expandafter{\\expanded{%s}}", data))
2061 data = texgettoks"mplibtmptoks"
```

Next line to address issue #55

```
2062 :gsub("###", "#")
2063 :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2064 end
2065 process(data, instancename)
2066 end
2067
```

`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.

```
2068 local figcontents = { post = { } }
2069 local function put2output(a,...)
2070     figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2071 end
2072 local function pdf_startfigure(n,llx,lly,urx,ury)
2073     put2output("\\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2074 end
2075 local function pdf_stopfigure()
2076     put2output("\\mplibstoptoPDF")
2077 end
```

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

```
2078 local function pdf_literalcode (...)
2079     put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2080 end
2081 local start_pdf_code = pdfmode
```

```

2082 and function() pdf_literalcode"q" end
2083 or function() put2output"\special{pdf:bcontent}" end
2084 local stop_pdf_code = pdfmode
2085 and function() pdf_literalcode"Q" end
2086 or function() put2output"\special{pdf:econtent}" end
2087

```

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

```

2088 local function put_tex_boxes (object,prescript)
2089   local box = prescript.mplibtexboxid:explode":"
2090   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2091   if n and tw and th then
2092     local op = object.path
2093     local first, second, fourth = op[1], op[2], op[4]
2094     local tx, ty = first.x_coord, first.y_coord
2095     local sx, rx, ry, sy = 1, 0, 0, 1
2096     if tw ~= 0 then
2097       sx = (second.x_coord - tx)/tw
2098       rx = (second.y_coord - ty)/tw
2099       if sx == 0 then sx = 0.00001 end
2100     end
2101     if th ~= 0 then
2102       sy = (fourth.y_coord - ty)/th
2103       ry = (fourth.x_coord - tx)/th
2104       if sy == 0 then sy = 0.00001 end
2105     end
2106

```

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

```

2107   local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2108   put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2109 end
2110 end
2111

```

Colors

```

2112 local function do_preobj_CR(object,prescript)
2113   if object.postscript == "collect" then return end
2114   local override = prescript and prescript.mpliboverridecolor
2115   if override then
2116     pdf_literalcode(override)
2117     if not pdfmode and override:find" [cC][sS] " then
2118       local name1 = override:match("^/(.-) cs ")
2119       local name2 = override:match(" /(.-) CS ")
2120       if name1 then
2121         texpstr(ccexplat, {
2122           "\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2123           name1, "}{\pdf_object_ref:n{" , name1, "}}"
2124         })
2125       end

```

```

2126     if name2 and name1 ~= name2 then
2127         texsprint(ccexplat, {
2128             "\\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2129             name2, "}\\pdf_object_ref:n{", name2, "}"
2130         })
2131     end
2132 end
2133 else
2134     local cs = object.color
2135     if cs and #cs > 0 then
2136         pdf_literalcode(luamplib.colorconverter(cs))
2137     end
2138 end
2139 end
2140

```

For transparency, shading, fading, and pattern

```

2141 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2142 local pdfobjs, pdfetcs = {}, {}
2143 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2144 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2145 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
2146 local update_pdfobjs
2147 if pdfmode then
2148     function update_pdfobjs (os, stream)
2149         local key = os
2150         if stream then key = key..stream end
2151         local on = key and pdfobjs[key]
2152         if on then
2153             return on, false
2154         end
2155         if stream then
2156             on = pdf.immediateobj("stream", stream, os)
2157         elseif os then
2158             on = pdf.immediateobj(os)
2159         else
2160             on = pdf.reserveobj()
2161         end
2162         if key then
2163             pdfobjs[key] = on
2164         end
2165         return on, true
2166     end
2167 else
2168     function update_pdfobjs (os, stream, hex)
2169         local key = os
2170         if stream then key = key..stream end
2171         local on = key and pdfobjs[key]
2172         if on then

```

```

2173     return on,false
2174 end
2175 on = pdfetcs.cnt or 1
2176 if stream then
2177     if hex then
2178         texsprintf(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2179     else
2180         texsprintf(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2181     end
2182 elseif os then
2183     texsprintf(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2184 else
2185     texsprintf(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2186 end
2187 pdfetcs.cnt = on + 1
2188 if key then
2189     pdfobjs[key] = on
2190 end
2191 return on,true
2192 end
2193 end
2194 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2195 if pdfmode then
2196     pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2197     local getpagers = pdfetcs.getpagers
2198     local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2199     local initialize_resources = function(name)
2200         local tabname = format("%s_res",name)
2201         pdfetcs[tabname] = { }
2202         if luatexbase.callbacktypes.finish_pdffile then -- ltuatex
2203             local obj = pdf.reserveobj()
2204             setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2205             luatexbase.add_to_callback("finish_pdffile", function()
2206                 pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
2207             end,
2208             format("luamplib.%s.finish_pdffile",name))
2209         end
2210     end
2211     pdfetcs.fallback_update_resources = function(name, res)
2212         local tabname = format("%s_res",name)
2213         if not pdfetcs[tabname] then
2214             initialize_resources(name)
2215         end
2216         if luatexbase.callbacktypes.finish_pdffile then
2217             local t = pdfetcs[tabname]
2218             t[#t+1] = res
2219         else
2220             local tpr, n = getpagers() or "", 0
2221             tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)

```

```

2222     if n == 0 then
2223         tpr = format("%s/%s<<%s>>", tpr, name, res)
2224     end
2225     setpagers(tpr)
2226 end
2227 end
2228 else
2229     texsprint {
2230         "\\luamplibatfirstshipout{",
2231         "\\special{pdf:obj @MPLibTr<<>>}",
2232         "\\special{pdf:obj @MPLibSh<<>>}",
2233         "\\special{pdf:obj @MPLibCS<<>>}",
2234         "\\special{pdf:obj @MPLibPt<<>>}}",
2235     }
2236     pdfetcs.fallback_update_resources = function (name,res,obj)
2237         texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2238         local tabname = format("%s_res",name)
2239         if not pdfetcs[tabname] then
2240             texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2241             pdfetcs[tabname] = { }
2242         end
2243         tableinsert(pdfetcs[tabname], res)
2244     end
2245 end
2246

```

Transparency

```

2247 local function add_extgs_resources (on, new)
2248     local key = format("MPLibTr%s", on)
2249     if new then
2250         local val = format(pdfetcs.resfmt, on)
2251         if pdfmanagement then
2252             texsprint {
2253                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2254             }
2255         else
2256             local tr = format("/%s %s", key, val)
2257             if is_defined(pdfetcs.pgfextgs) then
2258                 texsprint { "\\csname ", pdfetcs.pgfextgs, "\\endcsname{" , tr, "}" }
2259             elseif is_defined"TRP@list" then
2260                 texsprint(catat11,{
2261                     [[\if@files\immediate\write\auxout{]],
2262                     [[\string\g@addto@macro\string\TRP@list{]],
2263                     tr,
2264                     [[}]\fi]],
2265                 })
2266                 if not get_macro"TRP@list":find(tr) then
2267                     texsprint(catat11,[[\global\TRP@reruntrue]])
2268                 end

```

```

2269     else
2270         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2271     end
2272 end
2273 end
2274 return key
2275 end
2276
2277 local do_preobj_TR
2278 do
2279     local transparency_modes = {
2280         [0] = "Normal",
2281         "Normal",      "Multiply",      "Screen",      "Overlay",
2282         "SoftLight",    "HardLight",    "ColorDodge",  "ColorBurn",
2283         "Darken",       "Lighten",    "Difference",  "Exclusion",
2284         "Hue",          "Saturation", "Color",      "Luminosity",
2285         "Compatible",
2286         normal      = "Normal",    multiply   = "Multiply",    screen    = "Screen",
2287         overlay     = "Overlay",    softlight  = "SoftLight",   hardlight = "HardLight",
2288         colordodge  = "ColorDodge", colorburn  = "ColorBurn",   darken    = "Darken",
2289         lighten     = "Lighten",    difference = "Difference",  exclusion = "Exclusion",
2290         hue         = "Hue",        saturation = "Saturation",  color     = "Color",
2291         luminosity  = "Luminosity", compatible = "Compatible",
2292     }
2293     function do_preobj_TR(object,prescript)
2294         if object.postscript == "collect" then return end
2295         local opaq = prescript and prescript.tr_transparency
2296         if not opaq then return end
2297
2298         local key, on, os, new
2299         local mode = prescript.tr_alternative or 1
2300         mode = transparency_modes[tonumber(mode) or mode:lower()]
2301         if not mode then
2302             mode = prescript.tr_alternative
2303             warn("unsupported blend mode: '%s'", mode)
2304         end
2305         opaq = opaq:explode"."
2306         for i,v in ipairs(opaq) do
2307             opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2308         end
2309         for i,v in ipairs{ {mode,opaq[1],opaq[2] or opaq[1]},{"Normal",1,1} } do
2310             os = format("</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2311             on, new = update_pdfobjs(os)
2312             key = add_extgs_resources(on,new)
2313             if i == 1 then
2314                 pdf_literalcode("/%s gs",key)
2315             else
2316                 return format("/%s gs",key)
2317             end
2318         end
2319     end
2320 end

```

```

2318     end
2319 end
2320 end
2321

```

Shading with *metafun* format.

```

2322 local function add_shading_resources (on, new)
2323   if new then
2324     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2325     if pdfmanagement then
2326       texsprint {
2327         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2328       }
2329     else
2330       local res = format("/%s %s", key, val)
2331       pdfetcs.fallback_update_resources("Shading", res, "@MPlibSh")
2332     end
2333   end
2334 end
2335 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2336   for _, v in ipairs{ca, cb} do
2337     for i, vv in ipairs(v) do
2338       for ii, vvv in ipairs(vv) do
2339         v[i][ii] = tonumber(vvv) and format("%.3f", vvv) or vvv
2340       end
2341     end
2342   end
2343   local fun2fmt, os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2344   if steps > 1 then
2345     local list, bounds, encode = { }, { }, { }
2346     for i=1, steps do
2347       if i < steps then
2348         bounds[i] = format("%.3f", fractions[i] or 1)
2349       end
2350       encode[2*i-1] = 0
2351       encode[2*i] = 1
2352       os = fun2fmt:format(domain, tableconcat(ca[i], ' '), tableconcat(cb[i], ' '))
2353       :gsub(decimals, rmzeros)
2354       list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2355     end
2356     os = tableconcat {
2357       "<</FunctionType 3",
2358       format("/Bounds[%s]", tableconcat(bounds, ' ')),
2359       format("/Encode[%s]", tableconcat(encode, ' ')),
2360       format("/Functions[%s]", tableconcat(list, ' ')),
2361       format("/Domain[%s]>>", domain),
2362     } :gsub(decimals, rmzeros)
2363   else
2364     os = fun2fmt:format(domain, tableconcat(ca[1], ' '), tableconcat(cb[1], ' '))

```

```

2365 :gsub(decimals,rmzeros)
2366 end
2367 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2368 os = tableconcat {
2369   format("</ShadingType %i", shtype),
2370   format("/ColorSpace %s", colorspace),
2371   format("/Function %s", objref),
2372   format("/Coords[%s]", coordinates),
2373   format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2374 } :gsub(decimals,rmzeros)
2375 local on, new = update_pdfobjs(os)
2376 add_shading_resources(on, new)
2377 return on
2378 end
2379
2380 local function get_mp_matrix (matrix)
2381   process("mplibtransformmatrix(%s);"):format(matrix), "@mplibtransformmatrix")
2382   return luamplib.transformmatrix
2383 end
2384
2385 local do_preobj_SH
2386 do
2387   pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2388     run_tex_code({
2389       [[\color_model_new:nnn]],
2390       format("{mplibcolorspace_%s}", names:gsub(",","_")),
2391       format("{DeviceN}{names={%s}}", names),
2392       [[\mplibtmptoks\expanded{{\pdf_object_ref_last:}}]],
2393     }, ccexplat)
2394     local colorspace = texgettoks"mplibtmptoks"
2395     t[names] = colorspace
2396     return colorspace
2397   end })
2398   local function color_normalize(ca,cb)
2399     if #cb == 1 then
2400       if #ca == 4 then
2401         cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2402       else -- #ca = 3
2403         cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2404       end
2405     elseif #cb == 3 then -- #ca == 4 : rgb to cmyk
2406       local c, m, y = 1-cb[1], 1-cb[2], 1-cb[3]
2407       local k = math.min(c, m, y)
2408       if k == 1 then
2409         c, m, y = 0, 0, 0
2410       else
2411         c, m, y = (c-k)/(1-k), (m-k)/(1-k), (y-k)/(1-k)
2412       end
2413       cb[1], cb[2], cb[3], cb[4] = c, m, y, k

```

```

2414     end
2415 end
2416 local function write_mesh_objs (shtype, colorspace, stream, devicen, perrow)
2417     local cc = colorspace == "/DeviceCMYK" and 4
2418             or colorspace == "/DeviceRGB" and 3
2419             or devicen or 1
2420     local dict = format(
2421         "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2422         shtype,
2423         perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2424         colorspace,
2425         (" 0 1"):rep(cc))
2426     local on, new
2427     if pdfmode then
2428         on, new = update_pdfobjs(dict, stream)
2429     else
2430         stream = format("%02X"):rep(stream:len()), stream:byte(1,stream:len())
2431         on, new = update_pdfobjs(dict, stream, true) -- hex is true
2432     end
2433     add_shading_resources(on, new)
2434     return on
2435 end
2436 local function colors_ff (colors)
2437     local t, devicen = { }, { }
2438     for i,v in ipairs(colors) do
2439         t[i] = { }
2440         for _,vv in ipairs(v) do
2441             local tt = tableconcat(vv," "):explode()
2442             for _,vvv in ipairs(tt) do
2443                 tableinsert(t[i], string.char(math.floor(vvv * 0xFF)))
2444             end
2445             devicen = #tt
2446         end
2447     end
2448     return t, devicen
2449 end
2450 local function coords_ffff (coords)
2451     local Xt, Yt = { }, { }
2452     for i,v in ipairs(coords) do
2453         for ii,vv in ipairs(v) do
2454             local t = ii % 2 == 1 and Xt or Yt
2455             t[#t+1] = tonumber(vv)
2456         end
2457     end
2458     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2459     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2460     local wd, ht = xmax - xmin, ymax - ymin
2461     for i,v in ipairs(coords) do
2462         for ii,vv in ipairs(v) do

```

```

2463     local min = ii % 2 == 1 and xmin or ymin
2464     local dim = ii % 2 == 1 and wd or ht
2465     coords[i][ii] = string.pack(">H", math.floor((vv - min)/dim * 0xFFFF ))
2466     end
2467 end
2468 return coords, xmin, ymin, wd, ht
2469 end
2470 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2471     local perrow = prescript.sh_lattice_perrow or 2
2472     local steps = tonumber(prescript.sh_step) or 0
2473     local coords, colors = { }, { }
2474     if steps < 4 then
2475         local path = object.path
2476         for _,i in ipairs{1,2,4,3} do
2477             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2478         end
2479         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}}, {{1,1,0}} }
2480         colorspace = "/DeviceRGB"
2481     else
2482         local t = prescript.sh_lattice_data:explode()
2483         for i = 1, #t, 2 do
2484             coords[#coords+1] = { t[i], t[i+1] }
2485         end
2486         for i = 1, steps do
2487             if not ca[i] then err"colorspace mismatch?" end
2488             colors[#colors+1] = { ca[i] }
2489         end
2490     end
2491     if #coords % perrow ~= 0 then err"vertices_per_row mismatches lattice_coords" end
2492
2493     local stream, xmin, ymin, wd, ht, devicen = { }
2494     coords, xmin, ymin, wd, ht = coords_ffff(coords)
2495     colors, devicen = colors_ff(colors)
2496     for i = 1, #coords do
2497         stream[#stream+1] = tableconcat(coords[i])
2498         stream[#stream+1] = tableconcat(colors[i])
2499     end
2500
2501     local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2502     local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2503     return on, matrix
2504 end
2505 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2506     local steps = tonumber(prescript.sh_step) or 0
2507     local coords, colors, edges = { }, { }, { }
2508     if steps < 3 then
2509         local path = object.path
2510         for i = 1, 3 do
2511             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }

```

```

2512     end
2513     colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}} }
2514     edges = { 0, 0, 0 }
2515     colorspace = "/DeviceRGB"
2516 else
2517     for i = 1, steps do
2518         local t = prescript["sh_triangle_vertex_" .. i]:explode()
2519         if #t == 2 then
2520             coords[#coords+1] = { t[1], t[2] }
2521         elseif #t == 6 then
2522             coords[#coords+1] = { t[1], t[2] }
2523             coords[#coords+1] = { t[3], t[4] }
2524             coords[#coords+1] = { t[5], t[6] }
2525         end
2526         if not ca[i] then err"colorspace mismatch?" end
2527         colors[#colors+1] = { ca[i] }
2528         edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])
2529     end
2530 end
2531
2532 local stream, xmin, ymin, wd, ht, devicen = { }
2533 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2534 colors, devicen = colors_ff(colors)
2535 for i = 1, #coords do
2536     stream[#stream+1] = string.char(edges[i])
2537     stream[#stream+1] = tableconcat(coords[i])
2538     stream[#stream+1] = tableconcat(colors[i])
2539 end
2540
2541 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2542 local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2543 return on, matrix
2544 end
2545 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2546     local tensor = prescript.sh_type == "tensor"
2547     local steps = tonumber(prescript.sh_step) or 0
2548     local coords, colors, edges = { }, { }, { }
2549     if steps < 2 then
2550         local path, t = object.path, { }
2551         for i = 1, 4 do
2552             t[#t+1] = path[i].x_coord
2553             t[#t+1] = path[i].y_coord
2554             t[#t+1] = path[i].right_x
2555             t[#t+1] = path[i].right_y
2556             local j = i == 4 and 1 or i+1
2557             t[#t+1] = path[j].left_x
2558             t[#t+1] = path[j].left_y
2559         end
2560         if tensor then

```

```

2561     t = table.move(prescript.sh_tensor_path:explode(), 1, 8, #t+1, t)
2562 end
2563 coords = { t }
2564 colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2565 edges = { 0 }
2566 colorspace = "/DeviceRGB"
2567 else
2568   for i = 1, steps do
2569     local edge = tonumber(prescript["sh_coons_edge_"..i])
2570     if edge then
2571       edges[#edges+1] = edge
2572       coords[#coords+1] = prescript["sh_coons_path_"..i]:explode()
2573       if edge == 0 then
2574         if not (ca[i] and cb[i] and ca[i+1] and cb[i+1]) then err"colorspace mismatch?" end
2575         colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2576       else
2577         if not (ca[i] and cb[i]) then err"colorspace mismatch?" end
2578         colors[#colors+1] = { ca[i], cb[i] }
2579       end
2580     end
2581   end
2582 end
2583
2584 local stream, xmin, ymin, wd, ht, devicen = { }
2585 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2586 colors, devicen = colors_ff(colors)
2587 for i = 1, #coords do
2588   stream[#stream+1] = string.char(edges[i])
2589   stream[#stream+1] = tableconcat(coords[i])
2590   stream[#stream+1] = tableconcat(colors[i])
2591 end
2592
2593 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2594 local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2595 return on, matrix
2596 end
2597 function do_preobj_SH(object, prescript)
2598   local shade_no
2599   local sh_type = prescript and prescript.sh_type
2600   if not sh_type then return end
2601
2602   local domain = prescript.sh_domain or "0 1"
2603   local centera = (prescript.sh_center_a or "0 0"):explode()
2604   local centerb = (prescript.sh_center_b or "0 0"):explode()
2605   local transform = prescript.sh_transform == "yes"
2606   local sx,sy,sr,dx,dy = 1,1,1,0,0
2607   if transform then
2608     local first = (prescript.sh_first or "0 0"):explode()
2609     local setx = (prescript.sh_set_x or "0 0"):explode()

```

```

2610 local sety = (prescript.sh_set_y or "0 0"):explode()
2611 local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2612 if x ~= 0 and y ~= 0 then
2613     local path = object.path
2614     local path1x = path[1].x_coord
2615     local path1y = path[1].y_coord
2616     local path2x = path[x].x_coord
2617     local path2y = path[y].y_coord
2618     local dxa = path2x - path1x
2619     local dya = path2y - path1y
2620     local dx = setx[2] - first[1]
2621     local dy = sety[2] - first[2]
2622     if dxa ~= 0 and dya ~= 0 and dx ~= 0 and dy ~= 0 then
2623         sx = dxa / dx ; if sx < 0 then sx = - sx end
2624         sy = dya / dy ; if sy < 0 then sy = - sy end
2625         sr = math.sqrt(sx^2 + sy^2)
2626         dx = path1x - sx*first[1]
2627         dy = path1y - sy*first[2]
2628     end
2629 end
2630 end
2631 local ca, cb, colorspace, steps, fractions
2632 ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:" }
2633 cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:" }
2634 steps = tonumber(prescript.sh_step) or 1
2635 if steps > 1 then
2636     fractions = { prescript.sh_fraction_1 or 0 }
2637     for i=2,steps do
2638         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2639         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
2640         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
2641     end
2642 end
2643 if prescript.mplib_spotcolor then
2644     ca, cb = { }, { }
2645     local names, pos, objref = { }, -1, ""
2646     local script = object.prescript:explode"\13+"
2647     for i=#script,1,-1 do
2648         local name, value
2649         if script[i]:find"mplib_spotcolor" then
2650             local t = script[i]:explode"="[2]:explode:"
2651             value, objref, name = t[1], t[2], t[3]
2652         elseif script[i]:find"mplib_texcolor" then
2653             local t = script[i]:explode"="[2]:explode"!
2654             name, value = t[1], (t[2] or 100)/100
2655         end
2656         if name and value then
2657             if not names[name] then
2658                 pos = pos+1

```

```

2659         names[name] = pos
2660         names[#names+1] = name
2661     end
2662     local t = { }
2663     for j=1,names[name] do t[#t+1] = 0 end
2664     t[#t+1] = value
2665     tableinsert(#ca == #cb and ca or cb, t)
2666 end
2667 end
2668 for _,t in ipairs{ca,cb} do
2669     for _,tt in ipairs(t) do
2670         for i=1,#names-#tt do tt[#tt+1] = 0 end
2671     end
2672 end
2673 if #names == 1 then
2674     colorspace = objref
2675 else
2676     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2677 end
2678 else
2679     local model = 0
2680     for _,t in ipairs{ca,cb} do
2681         for _,tt in ipairs(t) do
2682             model = model > #tt and model or #tt
2683         end
2684     end
2685     for _,t in ipairs{ca,cb} do
2686         for _,tt in ipairs(t) do
2687             if #tt < model then
2688                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2689             end
2690         end
2691     end
2692     colorspace = model == 4 and "/DeviceCMYK"
2693                 or model == 3 and "/DeviceRGB"
2694                 or model == 1 and "/DeviceGray"
2695                 or err"unknown color model"
2696 end
2697 local extend = prescript.sh_extend
2698 local mesh_matrix
2699 if sh_type == "linear" then
2700     local coordinates = format("%f %f %f %f",
2701         dx + sx*centera[1], dy + sy*centera[2],
2702         dx + sx*centerb[1], dy + sy*centerb[2])
2703     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2704 elseif sh_type == "circular" then
2705     local factor = prescript.sh_factor or 1
2706     local radiusa = factor * prescript.sh_radius_a
2707     local radiusb = factor * prescript.sh_radius_b

```

```

2708     local coordinates = format("%f %f %f %f %f %f",
2709         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2710         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2711     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2712 elseif sh_type == "coons" or sh_type == "tensor" then
2713     shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2714 elseif sh_type == "triangle" then
2715     shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2716 elseif sh_type == "lattice" then
2717     shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2718 else
2719     err"unknown shading type"
2720 end
2721
2722 local matrix = prescript.sh_matrix
2723 if matrix and matrix:find"%a" then
2724     local t = get_mp_matrix(matrix)
2725     matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2726 end
2727 if mesh_matrix and matrix then
2728     local a, b = mesh_matrix:explode(), matrix:explode()
2729     matrix = format("%f %f %f %f %f %f",
2730         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],
2731         a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)
2732 end
2733
2734 return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix
2735 end
2736 end
2737

```

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

```

2738 local function add_pattern_resources (key, val)
2739     if pdfmanagement then
2740         texsprint {
2741             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2742         }
2743     else
2744         local res = format("/%s %s", key, val)
2745         if is_defined(pdfetcs.pgfpattern) then
2746             texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2747         else
2748             pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2749         end
2750     end
2751 end
2752 if pdfmode then
2753     function luamplib.dolatelua (on, os, matrix)
2754         local h, v = pdf.getpos()

```

```

2755     local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2756     matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2757     :gsub(decimals,rmzeros)
2758     pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2759     pdf.refobj(on)
2760 end
2761 else
2762 pdfetcs.shadingpatterns = { }
2763 pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2764 function luamplib.dolatelua (on, kind, xobj)
2765     local h, v
2766     local t = pdfetcs.shadingpatterns[on] or { }
2767     local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2768                 or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2769     if shift then
2770         h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2771     else
2772         h, v = pdf.getpos()
2773         h = format("%f", h/factor) :gsub(decimals,rmzeros)
2774         v = format("%f", v/factor) :gsub(decimals,rmzeros)
2775     end
2776     if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2777         warn"Rerun to get correct shading pattern"
2778     end
2779     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2780     local init = pdfetcs.shadingpatterninit_w
2781     if init then pdfetcs.shadingpatterninit_w = nil end
2782     local f = ioopen(name, init and "w" or "a")
2783     if f then
2784         f:write(("<<%s %s %s\n"):format(on, h, v))
2785         f:close()
2786     else
2787         err"cannot write a file. check the cache dir path"
2788     end
2789 end
2790 end
2791 local function do_preobj_shading (object, prescript)
2792     if not prescript or not prescript.sh_operand_type then return end
2793     local on,_,matrix = do_preobj_SH(object, prescript)
2794     local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2795     matrix = matrix or "1 0 0 1 0 0"
2796     if prescript.sh_in_xobj == "yes" then
2797         on = update_pdfobjs(("<<%s/Matrix[%s]>>"):format(os, matrix))
2798         goto skip_latelua
2799     end
2800     on = update_pdfobjs()
2801     if pdfmode then
2802         put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"]],[",matrix,"]])"})
2803     else

```

```

2804 local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2805         or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2806 local init = pdfetcs.shadingpatterninit_r
2807 if init then
2808     pdfetcs.shadingpatterninit_r = nil
2809     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2810     local f = ioopen(name)
2811     if f then
2812         for line in f:lines() do
2813             local t = line:explode()
2814             pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2815         end
2816         f:close()
2817     end
2818 end

```

This seems to be needed for proper functioning:

```

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

2819 local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2820 local mt = matrix:explode()
2821 matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2822 texsprint{ "\special{pdf:put ", format(pdfetcs.resfmt, on),
2823             format(" <<%s/Matrix[%s]>>}", os, matrix) }
2824 put2output("\latelua{ luamplib.dolatelua(%s,%s) }", on,
2825             xobj and ("'%s',[[%s]]"):format(xobj[1], xobj[2]))
2826 end
2827 ::skip_latelua::
2828 local key, val = format("MPlibPt%s", on), format(pdfetcs.resfmt, on)
2829 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.

```

2830 local stroke = prescript.sh_stroking
2831 if stroke == "filldraw" or stroke == "both" then
2832     pdf_literalcode("/Pattern cs/%s scn /Pattern CS/%s SCN", key, key)
2833 elseif stroke == "draw" or stroke == "yes" then
2834     pdf_literalcode("/Pattern CS/%s SCN", key)
2835 else
2836     pdf_literalcode("/Pattern cs/%s scn", key)
2837 end

```

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

```

2838 prescript.sh_type = nil
2839 end
2840

```

Tiling Patterns

```

2841 pdfetcs.patterns = { }
2842 local function gather_resources (optres, ispattern)
2843   local t = { }
2844   if pdfmanagement then
2845     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2846       local mytoks
2847       run_tex_code ( {
2848         "\\mplibmtokstoks\\expanded{ {",
2849         "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/", v, "}",
2850         "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/", v, "}}", " } }",
2851       }, ccexplat )
2852       mytoks = texgettoks "mplibmtokstoks"
2853       if not pdfmode then
2854         mytoks = mytoks:gsub("\\str_convert_pdfname:n{s*{(.-)})", "%1") -- why not expanded?
2855       end
2856       mytoks = mytoks and mytoks:gsub("^s*(.)s*$", "%1")
2857       if mytoks and mytoks ~= "" then
2858         t[#t+1] = ("/%s<<%s>>"):format(v, mytoks)
2859       end
2860     end
2861   elseif is_defined(pdfetcs.pgftextgs) then
2862     run_tex_code "\\relax" -- flush tex.sprint queue
2863     if pdfmode then
2864       for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2865                         ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2866                         Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2867         local res = (get_macro(v) or ""):gsub("^s*(.)s*$", "%1")
2868         if res ~= "" then
2869           t[#t+1] = ("/%s<<%s>>"):format(k, res )
2870         end
2871       end
2872     else
2873       local abc = get_macro "pgfutil@abc" or ""
2874       for k,v in pairs { ExtGState = "@pgftextgs",
2875                         ColorSpace = "@pgfcolorspaces",
2876                         Pattern = "@pgfpatterns", } do
2877         local tt = { }
2878         for vv in abc:gmatch( v .. "s*(%b<>)" ) do
2879           tt[#tt+1] = vv:match("^<<s*(.)s*>>$")
2880         end
2881         if #tt > 0 then
2882           t[#t+1] = ("/%s<<%s>>"):format(k, tableconcat(tt) )
2883         end
2884       end
2885     end
2886   end
2887 end

```

We still have to deal with Shading resources.

```

2886   if luatexbase.callbacktypes.finish_pdffile then
2887     if pdfetcs.Shading_res then

```

```

2888     t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2889   end
2890 else
2891   local res = pdfetcs.getpageres()
2892   res = res and res:match("/Shading%s*%b<>")
2893   if res then
2894     t[#t+1] = res
2895   end
2896 end
2897 else
2898   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

```

2899     warn"transparent package is not fully functional without pdfmanagement code."
2900   end
2901   if luatexbase.callbacktypes.finish_pdffile then
2902     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2903       local tt = pdfetcs[v.."_res"]
2904       if tt then
2905         t[#t+1] = ("/%s<<%s>>"):format(v, tableconcat(tt))
2906       end
2907     end
2908   else
2909     local res = pdfetcs.getpageres()
2910     if res then
2911       t[#t+1] = res
2912     end
2913   end
2914 end
2915 local result = tableconcat(t)
2916 if optres ~= "" then
2917   for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2918     local res = optres:match("/"..v.."s*%b<>")
2919     if res then
2920       if result:find("/"..v) then
2921         res = res:match("<<(.+)>>$")
2922         result = result:gsub("/"..v.."s*<<", "%1"..res, 1)
2923       else
2924         result = result .. res
2925       end
2926     end
2927   end
2928 end
2929 return result
2930 end
2931 function luamplib.registerpattern ( boxid, name, opts )
2932   local box = texgetbox(boxid)
2933   local wd = format("%.3f", box.width/factor)

```

```

2934 local hd = format("%.3f", (box.height+box.depth)/factor)
2935 info("w/h/d of pattern '%s': %s 0", name, format("%s %s", wd, hd):gsub(decimals,rmzeros))
2936 if opts.xstep == 0 then opts.xstep = nil end
2937 if opts.ystep == 0 then opts.ystep = nil end
2938 if opts.colored == nil then
2939   opts.colored = opts.coloured
2940   if opts.colored == nil then
2941     opts.colored = true
2942   end
2943 end
2944 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2945 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2946 if opts.matrix and opts.matrix:find"%a" then
2947   local t = get_mp_matrix(opts.matrix)
2948   opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2949   opts.xshift = opts.xshift or format("%f", t[5])
2950   opts.yshift = opts.yshift or format("%f", t[6])
2951 end
2952 local attr = {
2953   "/Type/Pattern",
2954   "/PatternType 1",
2955   format("/PaintType %i", opts.colored and 1 or 2),
2956   "/TilingType 2",
2957   format("/XStep %s", opts.xstep or wd),
2958   format("/YStep %s", opts.ystep or hd),
2959   format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2960 }
2961 local optres = opts.resources or ""
2962 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2963 local patterns = pdfetcs.patterns
2964 if pdfmode then
2965   if opts.bbox then
2966     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2967   end
2968   attr = tableconcat(attr) :gsub(decimals,rmzeros)
2969   local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2970   patterns[name] = { id = index, colored = opts.colored }
2971 else
2972   local cnt = #patterns + 1
2973   local objname = "@mplibpattern" .. cnt
2974   local metric = format("bbox %s", opts.bbox or format("0 0 %s %s", wd, hd))
2975   texsprint {
2976     "\\expandafter\\newbox\\cname luamplib.patternbox.", cnt, "\\endcsname",
2977     "\\global\\setbox\\cname luamplib.patternbox.", cnt, "\\endcsname",
2978     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2979     "\\special{pdf:bcontent}",
2980     "\\special{pdf:bxobj ", objname, " ", metric, "}",
2981     "\\raise\\dp\\cname luamplib.patternbox.", cnt, "\\endcsname",
2982     "\\box\\cname luamplib.patternbox.", cnt, "\\endcsname",

```

```

2983     "\\special{pdf:put @resources <<", optres, ">>}",
2984     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2985     "\\special{pdf:econtent}}",
2986   }
2987   patterns[cnt] = objname
2988   patterns[name] = { id = cnt, colored = opts.colored }
2989   patterns[name].shifts = { get_macro"MPlly", get_macro"MPlly" } -- for shading patterns above
2990 end
2991 end
2992
2993 local do_preobj_PAT
2994 do
2995   local function pattern_colorspace (cs)
2996     local on, new = update_pdfobjs(format("/Pattern %s]", cs))
2997     if new then
2998       local key, val = format("MPlibCS%i", on), format(pdfetcs.resfmt, on)
2999       if pdfmanagement then
3000         texsprintf {
3001           "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
3002         }
3003       else
3004         local res = format("/%s %s", key, val)
3005         if is_defined(pdfetcs.pgfcolorspace) then
3006           texsprintf { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
3007         else
3008           pdfetcs.fallback_update_resources("ColorSpace", res, "@MPlibCS")
3009         end
3010       end
3011     end
3012     return on
3013   end
3014   local function uncoloredGS(color, key, fill)
3015     local cs
3016     if color:find" cs " then
3017       local name
3018       if fill then
3019         name, color = color:match"^/(.-) cs (.-) sc"
3020       else
3021         name, color = color:match"/(.-) CS (.-) SC"
3022       end
3023       if pdfmode then
3024         cs = format("%s 0 R", ltx.pdf.object_id( name ))
3025         if cs == "0 0 R" then -- assumes colorspace.sty
3026           _, cs = get_spc_name_obj("/"..name)
3027         end
3028       else
3029         run_tex_code({ "\\mplibtmptoks\\expanded{\\pdf_object_ref:n{", name, "}}"} , ccexplat)
3030         cs = texgettoks"mplibtmptoks"
3031       end

```

```

3032     else
3033         if fill then
3034             color = colorsplit(color)
3035         else
3036             color = color:match"%a+ (.+) %a+":explode()
3037         end
3038         cs = #color == 4 and "/DeviceCMYK" or #color == 3 and "/DeviceRGB" or "/DeviceGray"
3039         color = tableconcat(color, " ")
3040     end
3041     return fill and format("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3042         or format("/MPLibCS%i CS %s /%s SCN", pattern_colorspace(cs), color, key)
3043 end
3044 function do_preobj_PAT(object, prescript)
3045     local name = prescript and prescript.mplibpattern
3046     if not name then return end
3047     local patterns = pdfetcs.patterns
3048     local patt = patterns[name]
3049     local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3050     local key = format("MPLibPt%s", index)
3051     local stroke, fillgs, strkgs = prescript.mplibpatternstroke
3052     if patt.colored then
3053         fillgs = format("/Pattern cs /%s scn", key)
3054         strkgs = stroke and format("/Pattern CS /%s SCN", key)
3055     else
3056         local color = prescript.mpliboverridecolor
3057         if not color then
3058             local t = object.color
3059             color = t and #t>0 and luamplib.colorconverter(t)
3060         end
3061         if not color then return end
3062         fillgs = uncoloredGS(color, key, true)
3063         strkgs = stroke and uncoloredGS(color, key, false)
3064     end
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.

```

3065     if stroke == "filldraw" or stroke == "both" then
3066         pdf_literalcode("%s %s", fillgs, strkgs)
3067     elseif stroke == "draw" or stroke == "yes" then
3068         pdf_literalcode(strkgs)
3069     else
3070         pdf_literalcode(fillgs)
3071     end
3072     if not patt.done then
3073         local val = pdfmode and format("%s 0 R", index) or patterns[index]
3074         add_pattern_resources(key, val)
3075     end
3076     patt.done = true
3077 end

```

```

3078 end
3079

```

Fading

```

3080 pdfetcs.fading = { }
3081 local function do_preobj_FADE (object, prescript)
3082   local fd_type = prescript and prescript.mplibfadetype
3083   local fd_stop = prescript and prescript.mplibfadestate
3084   if not fd_type then
3085     return fd_stop -- returns "stop" (if picture) or nil
3086   end
3087   local on, os, new
3088   if fd_type == "masking" then
3089     local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3090     on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
3091     local bc = prescript.mplibmaskingbgcolor
3092     bc = bc and bc:gsub(":", " ")
3093     bc = bc and ("BC[%s]"):format(bc):gsub(decimals, rmzeros) or ""
3094     os = format("<</SMask<</S/Luminosity/G %s%>>>",
3095                 pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3096   else
3097     local bbox = prescript.mplibfadebbox:explode":"
3098     local dx, dy = -bbox[1], -bbox[2]
3099     local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
3100     if not vec then
3101       if fd_type == "linear" then
3102         vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3103       else
3104         local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3105         vec = {centerx, centery, centerx, centery} -- center for both circles
3106       end
3107     end
3108     local coords = { vec[1], vec[2], vec[3], vec[4] }
3109     if fd_type == "linear" then
3110       coords = format("%f %f %f %f", tableunpack(coords))
3111     elseif fd_type == "circular" then
3112       local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3113       local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":"
3114       tableinsert(coords, 3, radius[1])
3115       tableinsert(coords, radius[2])
3116       coords = format("%f %f %f %f %f %f", tableunpack(coords))
3117     else
3118       err("unknown fading method '%s'", fd_type)
3119     end
3120     fd_type = fd_type == "linear" and 2 or 3
3121     local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3122     local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":" }
3123     local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":" }
3124     if steps > 1 then

```

```

3125     fractions = { prescript.sh_fraction_1 or 0 }
3126     for i=2,steps do
3127         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3128         ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":"
3129         cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":"
3130     end
3131 end
3132 local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3133 matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3134 matrix[5] = matrix[5] + dx
3135 matrix[6] = matrix[6] + dy
3136 matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3137 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3138 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3139 on = update_pdfobjs(os)
3140 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3141 local streamtext = format("q /Pattern cs/MPlibFd%s scn %s re f Q", on, bbox)
3142 :gsub(decimals,rmzeros)
3143 os = format("<</Pattern<</MPlibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3144 on = update_pdfobjs(os)
3145 local resources = format(pdfetcs.resfmt, on)
3146 on = update_pdfobjs("</S/Transparency/CS/DeviceGray>>")
3147 local attr = tableconcat{
3148     "/Subtype/Form",
3149     "/BBox[" .. bbox .. "]",
3150     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
3151     "/Resources " .. resources,
3152     "/Group " .. format(pdfetcs.resfmt, on),
3153 } :gsub(decimals,rmzeros)
3154 on = update_pdfobjs(attr, streamtext)
3155 os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
3156 end
3157 on, new = update_pdfobjs(os)
3158 local key = add_extgs_resources(on,new)
3159 start_pdf_code()
3160 pdf_literalcode("/%s gs", key)
3161 if fd_stop then return "standalone" end
3162 return "start"
3163 end
3164

```

Transparency Group

```

3165 pdfetcs.tr_group = { shifts = { } }
3166 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3167 local function do_preobj_GRP (object, prescript)
3168     local grstate = prescript and prescript.gr_state
3169     if not grstate then return end
3170     local trgroup = pdfetcs.tr_group
3171     if grstate == "start" then

```

```

3172   trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3173   trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3174   trgroup.wrapped = false
3175   for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+) do
3176     trgroup[v] = true
3177   end
3178   trgroup.bbox = prescript.mplibgroupbbox:explode":"
3179   put2output[[\begingroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3180 elseif grstate == "stop" then
3181   local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3182   put2output(tableconcat{
3183     "\\egroup",
3184     format("\\wd\\mplibscratchbox %fbp", urx-llx),
3185     format("\\ht\\mplibscratchbox %fbp", ury-lly),
3186     "\\dp\\mplibscratchbox 0pt",
3187   })
3188   local grattr
3189   if trgroup.off then
3190     grattr = ""
3191   else
3192     local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>",
3193                                     trgroup.isolated, trgroup.knockout))
3194     grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3195   end
3196   local res = gather_resources("")
3197   local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3198   if pdfmode then
3199     if trgroup.wrapped then
3200       put2output(tableconcat{
3201         "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3202         "/BBox[" .. bbox .. "]} resources{" .. res .. "}" .. "\\mplibscratchbox",
3203         "\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}",
3204       })
3205     end
3206     put2output(tableconcat{
3207       "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3208       "/BBox[" .. bbox .. "]", grattr, "}" .. resources{" .. res .. "}" .. "\\mplibscratchbox",
3209       "\\luamplibtagasgroupput{" .. trgroup.name .. "}" ..,
3210       [[\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}]],
3211       [[\\wd\\mplibscratchbox 0pt\\ht\\mplibscratchbox 0pt\\dp\\mplibscratchbox 0pt]],
3212       [[\\box\\mplibscratchbox]],
3213       "\\}\\endgroup",
3214       "\\\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\\\endcsname{" ..,
3215       "\\\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{" ..,
3216       "\\\\useboxresource \\the\\lastsavedboxresourceindex",
3217       "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3218       "\\\\box\\mplibscratchbox}",
3219     })
3220   else

```

```

3221 if trgroup.wrapped then
3222   trgroup.cnt = (trgroup.cnt or 0) + 1
3223   local objname = format("@mplibtrgr%s", trgroup.cnt)
3224   put2output(tableconcat{
3225     "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3226     "\\unhbox\\mplibscratchbox",
3227     "\\special{pdf:put @resources <<", res, ">>}",
3228     "\\special{pdf:exobj}",
3229     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3230   })
3231 end
3232 trgroup.cnt = (trgroup.cnt or 0) + 1
3233 local objname = format("@mplibtrgr%s", trgroup.cnt)
3234 put2output(tableconcat{
3235   "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3236   "\\unhbox\\mplibscratchbox",
3237   "\\special{pdf:put @resources <<", res, ">>}",
3238   "\\special{pdf:exobj <<", grattr, ">>}",
3239   "\\luamplibtagasgroupput{" .. trgroup.name .. "}",
3240   "\\special{pdf:uxobj ", objname, "}",
3241   "}\\endgroup",
3242 })
3243 token.set_macro("luamplib.group." .. trgroup.name, tableconcat{
3244   "\\setbox\\mplibscratchbox\\hbox{\\hskip", -llx, "bp\\raise", -lly, "bp\\hbox{",
3245   "\\special{pdf:uxobj ", objname, "}",
3246   "}}\\wd\\mplibscratchbox", urx-llx, "bp\\ht\\mplibscratchbox", ury-lly, "bp",
3247   "\\box\\mplibscratchbox",
3248   }, "global")
3249 end
3250 trgroup.shifts[trgroup.name] = { llx, lly }
3251 end
3252 return grstate
3253 end
3254 function luamplib.registergroup (boxid, name, opts)
3255   if opts.asgroup and opts.asgroup:find"wrapped" then
3256     luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3257     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
3258     opts.asgroup = opts.asgroup:gsub("wrapped", "")
3259   end
3260   local box = texgetbox(boxid)
3261   local wd, ht, dp = node.getwhd(box)
3262   local is_mask = opts.asgroup and opts.asgroup:find"masking"
3263   local res = opts.resources or ""
3264   res = gather_resources(res)
3265   local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3266   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
3267   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
3268   if opts.matrix and opts.matrix:find"%a" then
3269     local t = get_mp_matrix(opts.matrix)

```

```

3270     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3271 end
3272 local grtype = 3
3273 if opts.bbox then
3274     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3275     grtype = 2
3276 end
3277 local mpllx, mpily = get_macro'MPlly', get_macro'MPily'
3278 if is_mask then
3279     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3280     t[5], t[6] = t[5]+mpllx, t[6]+mpily
3281     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3282     mpllx, mpily = 0, 0
3283 end
3284 if opts.matrix then
3285     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3286     grtype = opts.bbox and 4 or 1
3287 end
3288 if opts.asgroup and not opts.asgroup:find"off" then
3289     local t = { isolated = false, knockout = false, masking = false }
3290     for _,v in ipairs(opts.asgroup:gsub("%s",""):explode",+") do t[v] = true end
3291     local on
3292     if t.masking then
3293         on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorsname or "/DeviceGray"))
3294     else
3295         local cs = opts.colorsname and ("/CS%s"):format(opts.colorsname) or ""
3296         on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3297     end
3298     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3299 end
3300 local trgroup = pdfetcs.tr_group
3301 trgroup.shifts[name] = { mpllx, mpily }
3302 local whd
3303 if pdfmode then
3304     attr = tableconcat(attr) :gsub(decimals,rmzeros)
3305     local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3306     token.set_macro("luamplib.group."..name, tableconcat{
3307         "\\useboxresource ", index,
3308     }, "global")
3309     whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3310 else
3311     trgroup.cnt = (trgroup.cnt or 0) + 1
3312     local objname = format("@mplibtrgr%s", trgroup.cnt)
3313     texsprint {
3314         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3315         "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3316         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3317         "\\special{pdf:bcontent}",
3318         "\\special{pdf:boxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",

```

```

3319     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3320     "\\special{pdf:put @resources <<", res, ">>}",
3321     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3322     "\\special{pdf:econtent}}",
3323   }
3324   token.set_macro("luamplib.group.".name, tableconcat{
3325     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3326     "\\wd\\mplibscratchbox ", wd, "sp",
3327     "\\ht\\mplibscratchbox ", ht, "sp",
3328     "\\dp\\mplibscratchbox ", dp, "sp",
3329     "\\box\\mplibscratchbox",
3330   }, "global")
3331   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3332   end
3333   info("w/h/d of group '%s': %s", name, whd)
3334 end
3335

```

luamplib.convert: flushing figures

```

3336 do
3337   local function stop_special_effects(fade,opaq)
3338     if fade then -- fading
3339       stop_pdf_code()
3340     end
3341     if opaq then -- opacity
3342       pdf_literalcode(opaq)
3343     end
3344   end
3345

```

For parsing prescript materials.

```

3346   local function script2table(s)
3347     local t = {}
3348     for _,i in ipairs(s:explode("\\13+")) do
3349       local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3350       if k and v and k ~= "" and not t[k] then
3351         t[k] = v
3352       end
3353     end
3354     return t
3355   end
3356

```

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

```

3357   local function pdf_textfigure(font,size,text,width,height,depth)
3358     text = text:gsub(".",function(c)
3359       return format("\\hbox{\\char%i}",string.byte(c)) -- kerning happens in metapost : false
3360     end)
3361     put2output("\\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)

```

```

3362 end
3363
3364 local bend_tolerance = 131/65536
3365
3366 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3367
3368 local function pen_characteristics(object)
3369     local t = mplib.pen_info(object)
3370     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3371     divider = sx*sy - rx*ry
3372     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3373 end
3374
3375 local function concat(px, py) -- no tx, ty here
3376     return (sy*px-ry*py)/divider, (sx*py-rx*px)/divider
3377 end
3378
3379 local function curved(ith,pth)
3380     local d = pth.left_x - ith.right_x
3381     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3382         abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3383         d = pth.left_y - ith.right_y
3384         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3385             abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3386             return false
3387         end
3388     end
3389     return true
3390 end
3391
3392 local function flushnormalpath(path,open)
3393     local pth, ith
3394     for i=1,#path do
3395         pth = path[i]
3396         if not ith then
3397             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3398         elseif curved(ith,pth) then
3399             pdf_literalcode("%f %f %f %f %f %f c",
3400                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3401         else
3402             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3403         end
3404         ith = pth
3405     end
3406     if not open then
3407         local one = path[1]
3408         if curved(pth,one) then
3409             pdf_literalcode("%f %f %f %f %f %f c",
3410                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )

```

```

3411     else
3412         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3413     end
3414 elseif #path == 1 then -- special case .. draw point
3415     local one = path[1]
3416     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3417 end
3418 end
3419
3420 local function flushconcatpath(path,open)
3421     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3422     local pth, ith
3423     for i=1,#path do
3424         pth = path[i]
3425         if not ith then
3426             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3427         elseif curved(ith,pth) then
3428             local a, b = concat(ith.right_x,ith.right_y)
3429             local c, d = concat(pth.left_x,pth.left_y)
3430             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3431         else
3432             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3433         end
3434         ith = pth
3435     end
3436     if not open then
3437         local one = path[1]
3438         if curved(pth,one) then
3439             local a, b = concat(pth.right_x,pth.right_y)
3440             local c, d = concat(one.left_x,one.left_y)
3441             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3442         else
3443             pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3444         end
3445     elseif #path == 1 then -- special case .. draw point
3446         local one = path[1]
3447         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3448     end
3449 end
3450

```

Finally, flush figures by inserting PDF literals.

```

3451 local function flush (result,flusher)
3452     if result then
3453         local figures = result.fig
3454         if figures then
3455             for f=1, #figures do
3456                 info("flushing figure %s",f)
3457                 local figure = figures[f]

```

```

3458     local objects = figure:objects()
3459     local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3460     local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3461     local bbox = figure:boundingbox()
3462     local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3463     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()

3464     else

```

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

```

3465     if tex_code_pre_mplib[f] then
3466         put2output(tex_code_pre_mplib[f])
3467     end
3468     pdf_startfigure(fignum,llx,lly,urx,ury)
3469     start_pdf_code()
3470     if objects then
3471         local savedpath = nil
3472         local savedhtap = nil
3473         for o=1,#objects do
3474             local object      = objects[o]
3475             local objecttype  = object.type

```

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

```

3476     local prescript      = object.prescript
3477     prescript = prescript and script2table(prescript) -- prescript is now a table
3478     do_preobj_CR(object,prescript) -- color
3479     local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3480     local fading_ = do_preobj_FADE(object,prescript) -- fading
3481     do_preobj_PAT(object,prescript) -- tiling pattern
3482     do_preobj_shading(object,prescript) -- shading pattern
3483     local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3484     if prescript and prescript.mplibtexboxid then
3485         put_tex_boxes(object,prescript)
3486     elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3487     elseif objecttype == "start_clip" then
3488         local evenodd = not object.istext and object.postscript == "evenodd"
3489         start_pdf_code()
3490         flushnormalpath(object.path,false)
3491         pdf_literalcode(evenodd and "W* n" or "W n")
3492     elseif objecttype == "stop_clip" then
3493         stop_pdf_code()
3494         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3495     elseif objecttype == "special" then

```

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

```

3496         if prescript and prescript.postmplibverbtx then
3497             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
3498         end
3499     elseif objecttype == "text" then
3500         local ot = object.transform -- 3,4,5,6,1,2
3501         start_pdf_code()
3502         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3503         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3504         stop_pdf_code()
3505     elseif not trgroup and fading_ ~= "stop" then
3506         local evenodd, collect, both = false, false, false
3507         local postscript = object.postscript
3508         if not object.istext then
3509             if postscript == "evenodd" then
3510                 evenodd = true
3511             elseif postscript == "collect" then
3512                 collect = true
3513             elseif postscript == "both" then
3514                 both = true
3515             elseif postscript == "eoboth" then
3516                 evenodd = true
3517                 both = true
3518             end
3519         end
3520         if collect then
3521             if not savedpath then
3522                 savedpath = { object.path or false }
3523                 savedhtap = { object.htap or false }
3524             else
3525                 savedpath[#savedpath+1] = object.path or false
3526                 savedhtap[#savedhtap+1] = object.htap or false
3527             end
3528         else

```

Removed from Con $\TeX$ t general: color stuff.

```

3529         local ml = object.miterlimit
3530         if ml and ml ~= miterlimit then
3531             miterlimit = ml
3532             pdf_literalcode("%f M",ml)
3533         end
3534         local lj = object.linejoin
3535         if lj and lj ~= linejoin then
3536             linejoin = lj
3537             pdf_literalcode("%i j",lj)
3538         end
3539         local lc = object.linecap
3540         if lc and lc ~= linecap then
3541             linecap = lc

```

```

3542         pdf_literalcode("%i J",lc)
3543     end
3544     local dl = object.dash
3545     if dl then
3546         local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3547         if d ~= dashed then
3548             dashed = d
3549             pdf_literalcode(dashed)
3550         end
3551     elseif dashed then
3552         pdf_literalcode("[] 0 d")
3553         dashed = false
3554     end
3555     local path = object.path
3556     local transformed, penwidth = false, 1
3557     local open = path and path[1].left_type and path[#path].right_type
3558     local pen = object.pen
3559     if pen then
3560         if pen.type == 'elliptical' then
3561             transformed, penwidth = pen_characteristics(object) -- boolean, value
3562             pdf_literalcode("%f w",penwidth)
3563             if objecttype == 'fill' then
3564                 objecttype = 'both'
3565             end
3566         else -- calculated by mplib itself
3567             objecttype = 'fill'
3568         end
3569     end
end

```

Added : shading

```

3570     local shade_no, shade_stroke, shade_cm = do_preobj_SH(object,prescript) -- shading
3571     if shade_no then
3572         pdf_literalcode"q /Pattern cs"
3573         objecttype = false
3574     end
3575     if transformed then
3576         start_pdf_code()
3577     end
3578     if path then
3579         if savedpath then
3580             for i=1,#savedpath do
3581                 local path = savedpath[i]
3582                 if transformed then
3583                     flushconcatpath(path,open)
3584                 else
3585                     flushnormalpath(path,open)
3586                 end
3587             end
3588             savedpath = nil

```

```

3589         end
3590         if transformed then
3591             flushconcatpath(path,open)
3592         else
3593             flushnormalpath(path,open)
3594         end
3595         if objecttype == "fill" then
3596             pdf_literalcode(evenodd and "h f*" or "h f")
3597         elseif objecttype == "outline" then
3598             if both then
3599                 pdf_literalcode(evenodd and "h B*" or "h B")
3600             else
3601                 pdf_literalcode(open and "S" or "h S")
3602             end
3603         elseif objecttype == "both" then
3604             pdf_literalcode(evenodd and "h B*" or "h B")
3605         end
3606     end
3607     if transformed then
3608         stop_pdf_code()
3609     end
3610     local path = object.htap

```

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

```

3611     if path then
3612         if transformed then
3613             start_pdf_code()
3614         end
3615         if savedhtap then
3616             for i=1,#savedhtap do
3617                 local path = savedhtap[i]
3618                 if transformed then
3619                     flushconcatpath(path,open)
3620                 else
3621                     flushnormalpath(path,open)
3622                 end
3623             end
3624             savedhtap = nil
3625             evenodd = true
3626         end
3627         if transformed then
3628             flushconcatpath(path,open)
3629         else
3630             flushnormalpath(path,open)
3631         end
3632         if objecttype == "fill" then
3633             pdf_literalcode(evenodd and "h f*" or "h f")
3634         elseif objecttype == "outline" then
3635             pdf_literalcode(open and "S" or "h S")

```

```

3636         elseif objecttype == "both" then
3637             pdf_literalcode(evenodd and "h B*" or "h B")
3638         end
3639         if transformed then
3640             stop_pdf_code()
3641         end
3642     end

```

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

```

3643         if shade_no then -- shading
3644             pdf_literalcode("W%s %s %s/MPlibSh%s sh Q",
3645                 evenodd and "*" or "",
3646                 shade_stroke and "s" or "n",
3647                 shade_cm and shade_cm.." cm " or "",
3648                 shade_no)
3649         end
3650     end
3651 end
3652 if fading_ == "start" then
3653     pdfetcs.fading.specialeffects = {fading_, tr_opaq}
3654 elseif trgroup == "start" then
3655     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq}
3656 elseif fading_ == "stop" then
3657     local se = pdfetcs.fading.specialeffects
3658     if se then stop_special_effects(se[1], se[2]) end
3659 elseif trgroup == "stop" then
3660     local se = pdfetcs.tr_group.specialeffects
3661     if se then stop_special_effects(se[1], se[2]) end
3662 else
3663     stop_special_effects(fading_, tr_opaq)
3664 end
3665 if fading_ or trgroup then -- extgs reseted
3666     miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3667 end
3668 end
3669 end
3670 stop_pdf_code()
3671 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimt_Ex code.

```

3672 for _,v in ipairs(figcontents) do
3673     if type(v) == "table" then
3674         texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3675     else
3676         texsprint(v)
3677     end
3678 end
3679 if #figcontents.post > 0 then texsprint(figcontents.post) end
3680 figcontents = { post = { } }
3681 end

```

```

3682     end
3683   end
3684 end
3685 end
3686
3687 function luamplib.convert (result, flusher)
3688   flush(result, flusher)
3689   return true -- done
3690 end
3691 end
3692
3693 function luamplib.colorconverter (cr)
3694   local n = #cr
3695   if n == 4 then
3696     local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3697     return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K", c,m,y,k,c,m,y,k), "0 g 0 G"
3698   elseif n == 3 then
3699     local r, g, b = cr[1], cr[2], cr[3]
3700     return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG", r,g,b,r,g,b), "0 g 0 G"
3701   else
3702     local s = cr[1]
3703     return format("%.3f g %.3f G", s,s), "0 g 0 G"
3704   end
3705 end

```

2.2 $\TeX$ package

First we need to load some packages.

```

3706 \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.

```

3707 \NeedsTeXFormat{LaTeX2e}
3708 \ProvidesPackage{luamplib}
3709   [2026/08/20 v2.42.8 mplib package for LuaTeX]
3710 \fi
3711 \ifdefined\newluafunction\else
3712   \input ltluatex
3713 \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.

```

3714 \ifnum\outputmode=0
3715   \ifdefined\AddToHookNext
3716     \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3717     \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3718     \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3719   \else

```

```

3720 \input atbegshi.sty
3721 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3722 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3723 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3724 \fi
3725 \fi

```

Loading of lua code.

```

3726 \directlua{require("luamplib")}

```

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

```

3727 \ifx\pdfoutput\undefined
3728 \let\pdfoutput\outputmode
3729 \fi
3730 \ifx\pdfliteral\undefined
3731 \protected\def\pdfliteral{\pdfextension literal}
3732 \fi

```

Set the format for METAPOST.

```

3733 \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.

```

3734 \ifnum\pdfoutput>0
3735 \let\mplibtoPDF\pdfliteral
3736 \else
3737 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3738 \ifcsname PackageInfo\endcsname
3739 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3740 \else
3741 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3742 \fi
3743 \fi

```

To make mplibcode typeset always in horizontal mode.

```

3744 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3745 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3746 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

3747 \def\mplibsetupcatcodes{%
3748 %catcode`\{=12 %catcode`\}=12
3749 \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3750 \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^M=12
3751 }

```

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

```

3752 \ifnum\outputmode>0 %
3753 \def\mplibputtextbox#1#2#3#4{%
3754 \pdfextension save\relax
3755 \vbox to 0pt{\vss

```

```

3756 \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
3757 \kern#3bp}%
3758 \pdfextension restore\relax}
3759 \else
3760 \def\mplibputtextbox#1#2#3#4{%
3761 \special{pdf:btrans matrix #4 #2 #3}%
3762 \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
3763 \special{pdf:etrans}}
3764 \fi

```

use Transparency Group

```

3765 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3766 \def\usemplibgroupmain#1{%
3767 \prependtomplibbox\hbox dir TLT\bgroup
3768 \csname luamplib.group.#1\endcsname
3769 \egroup
3770 }
3771 \protected\def\mplibgroup#1{%
3772 \begingroup
3773 \def\MPllx{0}\def\MPlly{0}%
3774 \def\mplibgroupname{#1}%
3775 \mplibgroupgetnexttok
3776 }
3777 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3778 \def\mplibgroupskipsspace{\afterassignment\mplibgroupgetnexttok\let\nexttok= }
3779 \def\mplibgroupbranch{%
3780 \ifx [\nexttok
3781 \expandafter\mplibgroupopts
3782 \else
3783 \ifx\mplibsptoken\nexttok
3784 \expandafter\expandafter\expandafter\mplibgroupskipsspace
3785 \else
3786 \let\mplibgroupoptions\empty
3787 \expandafter\expandafter\expandafter\mplibgroupmain
3788 \fi
3789 \fi
3790 }
3791 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3792 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3793 \protected\def\endmplibgroup{\egroup
3794 \directlua{ luamplib.registergroup(
3795 \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3796 )}%
3797 \endgroup
3798 }

```

Patterns

```

3799 {\def\:\global\let\mplibsptoken= } \: }
3800 \protected\def\mppattern#1{%
3801 \begingroup

```

```

3802 \def\MPllx{0}\def\MPlly{0}%
3803 \def\mplibpatternname{#1}%
3804 \mplibpatterngetnexttok
3805 }
3806 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3807 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3808 \def\mplibpatternbranch{%
3809   \ifx [\nexttok
3810     \expandafter\mplibpatternopts
3811   \else
3812     \ifx\mplibsptoken\nexttok
3813       \expandafter\expandafter\expandafter\mplibpatternskipsspace
3814     \else
3815       \let\mplibpatternoptions\empty
3816       \expandafter\expandafter\expandafter\mplibpatternmain
3817     \fi
3818   \fi
3819 }
3820 \def\mplibpatternopts[#1]{%
3821   \def\mplibpatternoptions{#1}%
3822   \mplibpatternmain
3823 }
3824 \def\mplibpatternmain{%
3825   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3826 }
3827 \protected\def\endmppattern{%
3828   \egroup
3829   \directlua{ luamplib.registerpattern(
3830     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3831   )}%
3832   \endgroup
3833 }

```

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

```

3834 \def\mpfiginstancename{@mpfig}
3835 \protected\def\mpfig{%
3836   \begingroup
3837   \futurelet\nexttok\mplibmpfigbranch
3838 }
3839 \def\mplibmpfigbranch{%
3840   \ifx *\nexttok
3841     \expandafter\mplibprempfig
3842   \else
3843     \ifx [\nexttok
3844       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3845     \else
3846       \expandafter\expandafter\expandafter\mplibmainmpfig
3847     \fi
3848   \fi

```

```

3849 }
3850 \def\mplibgobbleoptsmplibfig[#1]{\mplibmainmpfig}
3851 \def\mplibmainmpfig{%
3852   \begingroup
3853   \mplibsetupcatcodes
3854   \mplibdomainmpfig
3855 }
3856 \long\def\mplibdomainmpfig#1\endmpfig{%
3857   \endgroup
3858   \directlua{
3859     local legacy = luamplib.legacyverbatim
3860     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3861     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3862     luamplib.legacyverbatim = false
3863     luamplib.everymplib["\mpfiginstancename"] = ""
3864     luamplib.everyendmplib["\mpfiginstancename"] = ""
3865     luamplib.process_mplibcode(
3866       "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===].." "..everyendmpfig.." endfig;",
3867       "\mpfiginstancename")
3868     luamplib.legacyverbatim = legacy
3869     luamplib.everymplib["\mpfiginstancename"] = everympfig
3870     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3871   }%
3872   \endgroup
3873 }
3874 \def\mplibprempfig#1{%
3875   \begingroup
3876   \mplibsetupcatcodes
3877   \mplibdoprempfig
3878 }
3879 \long\def\mplibdoprempfig#1\endmpfig{%
3880   \endgroup
3881   \directlua{
3882     local legacy = luamplib.legacyverbatim
3883     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3884     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3885     luamplib.legacyverbatim = false
3886     luamplib.everymplib["\mpfiginstancename"] = ""
3887     luamplib.everyendmplib["\mpfiginstancename"] = ""
3888     luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
3889     luamplib.legacyverbatim = legacy
3890     luamplib.everymplib["\mpfiginstancename"] = everympfig
3891     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3892   }%
3893   \endgroup
3894 }
3895 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3896 \unless\ifcsname ver@luamplib.sty\endcsname
3897 \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3898 \protected\def\mplibcode{%
3899   \begingroup
3900   \futurelet\nexttok\mplibcodebranch
3901 }
3902 \def\mplibcodebranch{%
3903   \ifx [\nexttok
3904     \expandafter\mplibcodegetinstancename
3905   \else
3906     \global\let\currentmpinstancename\empty
3907     \expandafter\mplibcodeindeed
3908   \fi
3909 }
3910 \def\mplibcodeindeed{%
3911   \begingroup
3912   \mplibsetupcatcodes
3913   \mplibdocode
3914 }
3915 \long\def\mplibdocode#1\endmplibcode{%
3916   \endgroup
3917   \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]==],"\currentmpinstancename")}%
3918   \endgroup
3919 }
3920 \protected\def\endmplibcode{endmplibcode}
3921 \else

```

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

```

3922 \newenvironment{mplibcode}[1][{}]{%
3923   \xdef\currentmpinstancename{#1}%
3924   \mplibtmptoks{}\ltxdomplibcode
3925 }{}
3926 \def\ltxdomplibcode{%
3927   \begingroup
3928   \mplibsetupcatcodes
3929   \ltxdomplibcodeindeed
3930 }
3931 \def\mplib@mplibcode{mplibcode}
3932 \long\def\ltxdomplibcodeindeed#1\end#2{%
3933   \endgroup
3934   \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3935   \def\mplibtemp@a{#2}%
3936   \ifx\mplib@mplibcode\mplibtemp@a
3937     \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]==],"\currentmpinstancename")}%
3938     \end{mplibcode}%
3939   \else
3940     \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3941     \expandafter\ltxdomplibcode
3942   \fi

```

```

3943 }
3944 \fi

User settings.
3945 \def\mplibshowlog#1{\directlua{
3946   local s = string.lower("#1")
3947   if s == "enable" or s == "true" or s == "yes" then
3948     luampLib.showlog = true
3949   else
3950     luampLib.showlog = false
3951   end
3952 }}
3953 \def\mpliblegacybehavior#1{\directlua{
3954   local s = string.lower("#1")
3955   if s == "enable" or s == "true" or s == "yes" then
3956     luampLib.legacyverbatim = true
3957   else
3958     luampLib.legacyverbatim = false
3959   end
3960 }}
3961 \def\mplibverbatim#1{\directlua{
3962   local s = string.lower("#1")
3963   if s == "enable" or s == "true" or s == "yes" then
3964     luampLib.verbatiminput = true
3965   else
3966     luampLib.verbatiminput = false
3967   end
3968 }}
3969 \newtoks\mplibtmptoks

\everymplib & \everyendmplib: macros resetting luampLib.every(end)mpLib tables
3970 \ifcsname ver@luampLib.sty\endcsname
3971   \protected\def\everymplib{%
3972     \begingroup
3973     \mplibsetupcatcodes
3974     \mplibdoeverymplib
3975   }
3976   \protected\def\everyendmplib{%
3977     \begingroup
3978     \mplibsetupcatcodes
3979     \mplibdoeveryendmplib
3980   }
3981   \newcommand\mplibdoeverymplib[2][]{%
3982     \endgroup
3983     \directlua{
3984       luampLib.everymplib["#1"] = [==[\unexpanded{#2}]==]
3985     }%
3986   }
3987   \newcommand\mplibdoeveryendmplib[2][]{%
3988     \endgroup

```

```

3989 \directlua{
3990     luaplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===]
3991 }%
3992 }
3993 \else
3994 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3995 \protected\def\everymplib#1{%
3996     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3997     \begingroup
3998     \mplibsetupcatcodes
3999     \mplibdoeverymplib
4000 }
4001 \long\def\mplibdoeverymplib#1{%
4002     \endgroup
4003     \directlua{
4004         luaplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
4005     }%
4006 }
4007 \protected\def\everyendmplib#1{%
4008     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4009     \begingroup
4010     \mplibsetupcatcodes
4011     \mplibdoeveryendmplib
4012 }
4013 \long\def\mplibdoeveryendmplib#1{%
4014     \endgroup
4015     \directlua{
4016         luaplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
4017     }%
4018 }
4019 \fi

```

$\TeX$ macros for dimen/color

```

4020 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
4021 \def\mpcolor#1#{\domplibcolor{#1}}
4022 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

4023 \def\mplibnumbersystem#1{\directlua{
4024     local t = "#1"
4025     if t == "binary" then t = "decimal" end
4026     luaplib.numbersystem = t
4027 }}

```

Settings for .mp cache files.

```

4028 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4029 \def\mplibdomakenocache#1,{%
4030     \ifx\empty#1\empty
4031     \expandafter\mplibdomakenocache
4032     \else

```

```

4033 \ifx\stop#1\else
4034 \directlua{lua\plib.noneedtoreplace["#1.mp"]=true}%
4035 \expandafter\expandafter\expandafter\mplibdomakenocache
4036 \fi
4037 \fi
4038 }
4039 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4040 \def\mplibdocancelnocache#1,{%
4041 \ifx\empty#1\empty
4042 \expandafter\mplibdocancelnocache
4043 \else
4044 \ifx\stop#1\else
4045 \directlua{lua\plib.noneedtoreplace["#1.mp"]=false}%
4046 \expandafter\expandafter\expandafter\mplibdocancelnocache
4047 \fi
4048 \fi
4049 }
4050 \def\mplibcachedir#1{\directlua{lua\plib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4051 \def\mplibtexttextlabel#1{\directlua{
4052 local s = string.lower("#1")
4053 if s == "enable" or s == "true" or s == "yes" then
4054 lua\plib.texttextlabel = true
4055 else
4056 lua\plib.texttextlabel = false
4057 end
4058 }}
4059 \def\mplibcodeinherit#1{\directlua{
4060 local s = string.lower("#1")
4061 if s == "enable" or s == "true" or s == "yes" then
4062 lua\plib.codeinherit = true
4063 else
4064 lua\plib.codeinherit = false
4065 end
4066 }}
4067 \def\mplibglobaltexttext#1{\directlua{
4068 local s = string.lower("#1")
4069 if s == "enable" or s == "true" or s == "yes" then
4070 lua\plib.globaltexttext = true
4071 else
4072 lua\plib.globaltexttext = false
4073 end
4074 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

4075 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4076 \def\mplibstarttoPDF#1#2#3#4{%
4077   \prependtomplibbox
4078   \hbox dir TLT\bgroup
4079   \xdef\MPllx{#1}\xdef\MPlly{#2}%
4080   \xdef\MPurx{#3}\xdef\MPury{#4}%
4081   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4082   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4083   \parskip0pt%
4084   \leftskip0pt%
4085   \parindent0pt%
4086   \everypar{}%
4087   \setbox\mplibscratchbox\vbox\bgroup
4088   \noindent
4089 }
4090 \def\mplibstoptoPDF{%
4091   \par
4092   \egroup %
4093   \setbox\mplibscratchbox\hbox %
4094     {\hskip-\MPllx bp%
4095      \raise-\MPlly bp%
4096      \box\mplibscratchbox}%
4097   \setbox\mplibscratchbox\vbox to \MPheight
4098     {\vfill
4099      \hsize\MPwidth
4100      \wd\mplibscratchbox0pt%
4101      \ht\mplibscratchbox0pt%
4102      \dp\mplibscratchbox0pt%
4103      \box\mplibscratchbox}%
4104   \wd\mplibscratchbox\MPwidth
4105   \ht\mplibscratchbox\MPheight
4106   \box\mplibscratchbox
4107   \egroup
4108 }

```

Text items have a special handler.

```

4109 \def\mplibtexttext#1#2#3#4#5{%
4110   \begingroup
4111   \setbox\mplibscratchbox\hbox
4112     {\font\temp=#1 at #2bp%
4113      \temp
4114      #3}%
4115   \setbox\mplibscratchbox\hbox
4116     {\hskip#4 bp%
4117      \raise#5 bp%
4118      \box\mplibscratchbox}%
4119   \wd\mplibscratchbox0pt%
4120   \ht\mplibscratchbox0pt%
4121   \dp\mplibscratchbox0pt%
4122   \box\mplibscratchbox

```

```
4123 \endgroup
4124 }
```

Input luamplib.cfg when it exists.

```
4125 \openin0=luamplib.cfg
4126 \ifeof0 \else
4127 \closein0
4128 \input luamplib.cfg
4129 \fi
```

Code for tagpdf

```
4130 \def\luamplibtagtextboxset#1#2{#2}
4131 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4132 \let\luamplibtagasgroupset\relax
4133 \let\luamplibtagasgroupput\luamplibtagtextboxset
4134 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4135 \ifcsname ver@tagpdf.sty\endcsname \else
4136 \ExplSyntaxOn
4137 \keys_define:nn{luamplib/tagging}
4138 {
4139 ,alt .code:n = { }
4140 ,actualtext .code:n = { }
4141 ,artifact .code:n = { }
4142 ,text .code:n = { }
4143 ,off .code:n = { }
4144 ,tag .code:n = { }
4145 ,adjust-BBox .code:n = { }
4146 ,tagging-setup .code:n = { }
4147 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4148 ,instancename .meta:n = { instance = {#1} }
4149 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4150 }
4151 \RenewDocumentCommand\mplibcode{0{}}
4152 {
4153 \tl_gclear:N \currentmpinstancename
4154 \keys_set:ne{luamplib/tagging}{#1}
4155 \mplibtmptoks{}\ltxdomplibcode
4156 }
4157 \cs_set_eq:NN \mplibaltext \use_none:n
4158 \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.

```
4159 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4160 \ExplSyntaxOff
4161 \endinput\fi
4162 \ExplSyntaxOn
```

```

4163 \tl_new:N \l__luamplib_tag_envname_tl
4164 \tl_new:N \l__luamplib_tag_alt_tl
4165 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4166 \tl_new:N \l__luamplib_tag_actual_tl
4167 \tl_new:N \l__luamplib_tag_struct_tl
4168 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4169 \bool_new:N \l__luamplib_tag_usetext_bool
4170 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4171 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4172 \tl_new:N \l__luamplib_tag_bbox_draw_tl
4173 \tl_new:N \l__luamplib_BBox_llx_tl
4174 \tl_new:N \l__luamplib_BBox_lly_tl
4175 \tl_new:N \l__luamplib_BBox_urx_tl
4176 \tl_new:N \l__luamplib_BBox_ury_tl
4177 \msg_new:nnn {luamplib}{figure-text-reuse}
4178 {
4179   tex-text~box~#1~probably~is~incorrectly~tagged.~
4180   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4181   Check~the~resulting~PDF.
4182 }
4183 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4184 {
4185   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4186   Using~mplibgroup~with~text~mode~is~not~recommended.~
4187   Check~the~resulting~PDF.
4188 }
4189 \msg_new:nnn {luamplib}{alt-text-missing}
4190 {
4191   Alternate~text~for~#1~is~missing.~
4192   Using~the~default~value~'#2'~instead.
4193 }

```

Sockets for tex-text boxes.

```

4194 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4195 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4196 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4197 {

```

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.

```

4198 \bool_if:NTF \l__luamplib_tag_usetext_bool
4199 {
4200   \tag_mc_end_push:
4201   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4202   \cs_gset_nopar:cpe {luamplib.taggedbox.#1} {\tag_get:n{struct_num}}

```

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

```

4203   \tag_mc_begin:n{tag=text}
4204   #2
4205   \tag_mc_end:

```

```

4206 \tag_struct_end:
4207 \tag_mc_begin_pop:n{}
4208 }
4209 {
4210 \tag_suspend:n{\luamplibtagtextboxset}
4211 #2
4212 \tag_resume:n{\luamplibtagtextboxset}
4213 }
4214 }
4215 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4216 {
4217 \bool_lazy_and:nnTF
4218 { \l__luamplib_tag_usetext_bool }
4219 { \cs_if_free_p:c {luamplib.taggedbox.#1} }
4220 {
4221 \tag_resume:n{\mplibputtextbox}
4222 \tag_mc_end:
4223 \cs_if_exist:cTF {luamplib.taggedbox.#1}
4224 {
4225 \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4226 #2
4227 \cs_undefine:c {luamplib.taggedbox.#1}
4228 }
4229 {
4230 \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
4231 \tag_mc_begin:n{}
4232 \int_set:Nn \l_tmpa_int {#1}
4233 \tag_mc_reset_box:N \l_tmpa_int
4234 #2
4235 \tag_mc_end:
4236 }
4237 \tag_mc_begin:n{artifact}
4238 }
4239 {
4240 \int_set:Nn \l_tmpa_int {#1}
4241 \tag_mc_reset_box:N \l_tmpa_int
4242 #2
4243 }
4244 }
4245 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4246 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4247 \cs_set_nopar:Npn \luamplibtagtextboxset
4248 {
4249 \tag_socket_use:nnn{luamplib/texttext/set}
4250 }

```

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

```

4251 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2

```

```

4252 {
4253   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
4254   \bool_set_false:N \l__luamplib_tag_usetext_bool
4255   \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
4256   \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4257   \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4258 }
4259 \sys_if_output_pdf:TF
4260 {
4261   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4262   {
4263     \pdfextension save\relax
4264     \vbox to 0pt{\vss
4265       \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4266         \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4267       \kern #3bp}
4268     \pdfextension restore\relax
4269   }
4270 }
4271 {
4272   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4273   {
4274     \special{pdf:btrans~matrix~#4~#2~#3}
4275     \vbox to 0pt{\vss\hbox bdir0 to 0pt{
4276       \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4277     \special{pdf:etrans}
4278   }
4279 }

```

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

```

4280 \cs_set_nopar:Npn \luamplibtagasgroupset
4281 {
4282   \bool_set_false:N \l__luamplib_tag_usetext_bool
4283 }
4284 \cs_set_nopar:Npn \luamplibtagasgroupput
4285 {
4286   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4287   \tag_socket_use:nnn{luamplib/mplibgroup/put}
4288 }

```

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

```

4289 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4290 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4291 {
4292   \cs_if_free:cT {luamplib.mplibgroup.text.#1}
4293   {
4294     \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4295     \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4296   }

```

```

4297 \tag_mc_end:
4298 \tag_mc_begin:n{tag=text}
4299 #2
4300 \tag_mc_end:
4301 \tag_mc_begin:n{artifact}
4302 }
4303 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

A macro for BBox attribute

```

4304 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4305 {
4306   \tl_set:Nx \l_tmpa_tl {\luamplib.BBox.\tag_get:n{struct_num}}
4307   \tex_savepos:D
4308   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4309   \tl_set:Nx \l__luamplib_BBox_llx_tl
4310     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4311   \tl_set:Nx \l__luamplib_BBox_lly_tl
4312     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4313   \tl_set:Nx \l__luamplib_BBox_urx_tl
4314     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4315   \tl_set:Nx \l__luamplib_BBox_ury_tl
4316     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4317   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4318   {
4319     \int_zero:N \l_tmpa_int
4320     \tl_map_inline:nn
4321     {
4322       \l__luamplib_BBox_llx_tl
4323       \l__luamplib_BBox_lly_tl
4324       \l__luamplib_BBox_urx_tl
4325       \l__luamplib_BBox_ury_tl
4326     }
4327     {
4328       \int_incr:N \l_tmpa_int
4329       \tl_set:Nx ##1
4330       {
4331         \fp_eval:n
4332         {
4333           ##1
4334           +
4335           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4336         }
4337       }
4338     }
4339   }
4340   \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4341   {
4342     /O /Layout /BBox [
4343     \l__luamplib_BBox_llx_tl\c_space_tl

```

```

4344     \l__luamplib_BBox_lly_tl\c_space_tl
4345     \l__luamplib_BBox_urx_tl\c_space_tl
4346     \l__luamplib_BBox_ury_tl
4347   ]
4348 }
4349 \bool_if:NT \l__tag_graphic_debug_bool
4350 {
4351   \iow_log:e
4352   {
4353     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4354     \l__luamplib_BBox_llx_tl\c_space_tl
4355     \l__luamplib_BBox_lly_tl\c_space_tl
4356     \l__luamplib_BBox_urx_tl\c_space_tl
4357     \l__luamplib_BBox_ury_tl
4358   }
4359   \sys_if_output_pdf:TF
4360   {
4361     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
4362     {
4363       \pdfextension save\relax
4364       \opacity_select:n{0.5} \color_select:n{red}
4365       \pdfextension literal~text
4366       {
4367         \l__luamplib_BBox_llx_tl\c_space_tl
4368         \l__luamplib_BBox_lly_tl\c_space_tl
4369         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4370         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4371         re~f
4372       }
4373       \pdfextension restore\relax
4374     }
4375   }
4376   {
4377     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
4378     {
4379       \special{pdf:bcontent}
4380       \opacity_select:n{0.5} \color_select:n{red}
4381       \special{pdf:code~
4382         1~0~0~1~
4383         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4384         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4385         cm
4386       }
4387       \special{pdf:code~
4388         \l__luamplib_BBox_llx_tl\c_space_tl
4389         \l__luamplib_BBox_lly_tl\c_space_tl
4390         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4391         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4392         re~f

```

```

4393     }
4394     \special{pdf:econtent}
4395   }
4396 }
4397 }
4398 }

```

Sockets for main process

```

4399 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4400 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4401 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4402 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4403 {
4404   \tag_mc_end_push:
4405   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4406   {
4407     \tl_if_empty:eTF{#1}
4408     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4409     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4410     \msg_warning:nnVV{luamplib}{alt-text-missing}
4411     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4412   }
4413   \tag_struct_begin:n
4414   {
4415     tag=\l__luamplib_tag_struct_tl,
4416     alt=\l__luamplib_tag_alt_tl,
4417   }
4418   \tag_mc_begin:n{}
4419 }
4420 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4421 {
4422   \__luamplib_tag_bbox_attribute:n {#1}
4423   #2
4424   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4425   \tag_mc_end:
4426   \tag_struct_end:
4427   \tag_mc_begin_pop:n{}
4428 }
4429 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4430 {
4431   \tag_mc_end_push:
4432   \tag_struct_begin:n
4433   {
4434     tag=Span,
4435     actualtext=\l__luamplib_tag_actual_tl,
4436   }
4437   \tag_mc_begin:n{}
4438 }
4439 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}

```

```

4440 {
4441     #2
4442     \tag_mc_end:
4443     \tag_struct_end:
4444     \tag_mc_begin_pop:n{ }
4445 }
4446 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4447 {
4448     \tag_mc_end_push:
4449     \tag_mc_begin:n{artifact}
4450 }
4451 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4452 {
4453     #2
4454     \tag_mc_end:
4455     \tag_mc_begin_pop:n{ }
4456 }

```

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

```

4457 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4458 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4459 {
4460     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4461     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4462 }
4463 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4464 {
4465     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4466     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4467 \prependtomplibbox \mplibnoforcehmode
4468 \mode_if_vertical:T { \noindent \aftergroup\par }
4469 }
4470 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4471 {
4472     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4473     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4474 }
4475 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4476 {
4477     \bool_set_true:N \l__luamplib_tag_usetext_bool
4478     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4479     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4480 \prependtomplibbox \mplibnoforcehmode
4481 \mode_if_vertical:T { \noindent \aftergroup\par }
4482 }
4483 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4484 {

```

```

4485 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4486 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4487 }
4488 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```

4489 \keys_define:nn{luamplib/tagging}
4490 {
4491   ,alt .code:n =
4492   {
4493     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4494     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4495   }
4496   ,actualtext .code:n =
4497   {
4498     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4499     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4500   }
4501   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4502   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4503   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4504   ,tag .code:n =
4505   {
4506     \str_case:nnF {#1}
4507     {
4508       {false} { \keys_set:nn {luamplib/tagging} {off} }
4509       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4510     }
4511     {
4512       \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4513       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4514     }
4515   }
4516   ,adjust-BBox .code:n =
4517   {
4518     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4519     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4520   }
4521   ,tagging-setup .code:n = { \keys_set:known:nn {luamplib/tagging} {#1} }
4522 }
4523 \keys_define:nn {luamplib/instance}
4524 {
4525   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4526   ,instancename .meta:n = { instance = {#1} }
4527   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4528 }

```

Redefine our macros

```

4529 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4530 {

```

```

4531 \prependtomplibbox
4532 \hbox dir~TLT\bgroup
4533 \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4534 \xdef\MPllx{#1}\xdef\MPlly{#2}%
4535 \xdef\MPurx{#3}\xdef\MPury{#4}%
4536 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4537 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4538 \parskip0pt
4539 \leftskip0pt
4540 \parindent0pt
4541 \everypar{}%
4542 \setbox\mplibscratchbox\vbox\bgroup
4543 \tag_suspend:n{\mplibstarttoPDF}
4544 \noindent
4545 }
4546 \cs_set_nopar:Npn \mplibstoptoPDF
4547 {
4548 \par
4549 \egroup
4550 \setbox\mplibscratchbox\hbox
4551 {\hskip-\MPllx bp
4552 \raise-\MPlly bp
4553 \box\mplibscratchbox}%
4554 \setbox\mplibscratchbox\vbox to \MPheight
4555 {\vfill
4556 \hsize\MPwidth
4557 \wd\mplibscratchbox0pt
4558 \ht\mplibscratchbox0pt
4559 \dp\mplibscratchbox0pt
4560 \box\mplibscratchbox}%
4561 \wd\mplibscratchbox\MPwidth
4562 \ht\mplibscratchbox\MPheight
4563 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4564 \egroup
4565 }
4566 \RenewDocumentCommand\mplibcode{0{}}
4567 {
4568 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4569 \tl_gclear:N \currentmpinstancename
4570 \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4571 \keys_set:nV {luamplib/instance} \l_tmpa_tl
4572 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4573 \tag_socket_use:n{luamplib/figure/init}
4574 \mplibtmp toks{}\ltxdomplibcode
4575 }
4576 \RenewDocumentCommand\mpfig{s 0{}}
4577 {
4578 \begingroup
4579 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}

```

```

4580 \keys_set_known:ne {luamplib/tagging} {#2}
4581 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4582 \tag_socket_use:n{luamplib/figure/init}
4583 \IfBooleanTF{#1} { \mplibprempfig * }
4584             { \mplibmainmpfig }
4585 }
4586 \RenewDocumentCommand\usemplibgroup{0{} m}
4587 {
4588   \begingroup
4589   \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4590   \keys_set_known:ne {luamplib/tagging} {#1}
4591   \tag_socket_use:n{luamplib/figure/init}
4592   \prependtomplibbox\hbox dir~TLT\bgroup
4593     \tag_socket_use:nn{luamplib/figure/begin}{#2}
4594     \setbox\mplibscratchbox\hbox\bgroup
4595       \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
4596       \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4597       \egroup
4598       \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4599   \egroup
4600   \endgroup
4601 }

```

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

```

4602 \cs_new_nopar:Npn \mplibalttext #1
4603 {
4604   \tl_set:Nn \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4605 }
4606 \cs_new_nopar:Npn \mplibactualtext #1
4607 {
4608   \tl_set:Nn \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4609 }
4610 \ExplSyntaxOff

```

That's all folks!

3 The GNU GPL License v2

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

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

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

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

Preamble

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

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

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

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

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

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or 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.

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

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

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

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

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

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

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

NO WARRANTY

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

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

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

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

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

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

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

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

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

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

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

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

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

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

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

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