

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/08/07 v2.42.5

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	21
1.2.16	Form XObject from T _E X side	24
1.2.17	Masking	25
1.2.18	Free-form Gouraud-shaded triangle mesh shading	26
1.2.19	Lattice-form Gouraud-shaded triangle mesh shading	28
1.2.20	Coons patch mesh shading	28
1.2.21	Tensor-product patch mesh shading	30
1.3	Lua	31
1.3.1	runscript	31
1.3.2	Accessing METAPOST variables	32
1.3.3	Running a METAPOST code	32
1.3.4	Registering a tiling pattern	33
1.3.5	Registering a form XObject	33
2	Implementation	34
2.1	Lua module	34
2.2	T _E Xpackage	113
3	The GNU GPL License v2	134

1 Documentation

This package aims at providing a simple way to typeset directly METAPOST code in a document with LuaT_EX. LuaT_EX is built with the Lua mplib library, that runs METAPOST code. This package is basically a wrapper for the Lua mplib functions and some T_EX functions to have the output of the mplib functions in the PDF.

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

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

The code of luamplib is basically from the `luatex-mp lib.lua` and `luatex-mp lib.tex` files from ConT_EXt. They have been adapted to L^AT_EX and Plain by Elie Roux and Philipp Gesang and new functionalities have been added by Kim Dohyun. The most notable changes are:

- Possibility to use `btex ... etex` to typeset $\TeX$ code. `texttext <string>` is a more versatile macro equivalent to `TEX <string>` from `TEX.mp`. `TEX` is also allowed and is a synonym of `texttext`. The argument of `mplib`'s primitive `maketext` will also be processed by the same routine.
- Possibility to use `verbatimtex ... etex` to run a $\TeX$ code. `VerbatimTeX <string>` is a more versatile macro corresponding to `verbatimtex` command. Of course the behavior cannot be the same as the stand-alone `mpost`, so that you cannot include `\documentclass`, `\usepackage` etc. When these $\TeX$ commands are found in `verbatimtex ... etex`, the entire code will be ignored.

The treatment of `verbatimtex` command has changed a lot since v2.20: see below § 1.1.6.

- In the past, the package required PDF mode in order to have some output. Starting with v2.7 it works in DVI mode as well, though `DVIPDFMx` is the only DVI tool currently supported.

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

1.1 $\TeX$

1.1.1 Forcing horizontal mode

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

1.1.2 Insert some code into every code chunk

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

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



1.1.3 Choosing a METAPOST format

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

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

N.B. As *metafun* is such a complicated format, we cannot support all the special effects provided by *metafun*. At least, however, transparency (actually opacity), shading (gradient colors) and transparency group are fully supported, and outlinetext is supported by our own alternative `mpliboutlinetext` (see below § 1.2.9). You can try other effects as well, though we did not fully test their proper functioning.

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

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

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

From v2.36, shading is available with *plain* format as well with extended functionality. See below § 1.2.12.

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

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

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

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

1.1.4 Choosing a number system

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

1.1.5 Showing METAPOST log

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

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

1.1.6 verbatimtex Legacy behavior

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

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

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

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

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



diameter: 22.62764bp.

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

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

ABC **DEF GHI**

1.1.7 $\TeX$ -text labels

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

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

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

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

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

1.1.8 Inherit `METAPOST` code

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

1.1.9 `\mplibglobaltexttext{enable|disable}`

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

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



1.1.10 Separate `METAPOST` instances

`luamplib` v2.22 has added the support for several named `METAPOST` instances in $\TeX$ environment `mplibcode` or Plain $\TeX$ commands `\mplibcode` ... `\endmplibcode`. The syntax for $\TeX$ is:

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

The behavior is as follows.

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

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

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

1.1.11 Verbatim input

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

1.1.12 $\TeX$ `dimen`

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

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



1.1.13 $\TeX$ color

With the command `\mpcolor[...]{...}`, color expressions of `color`, `xcolor` and `l3color` module/packages can be used in the `mplibcode` environment (after `withcolor` command, in principle). See the example above at § 1.1.12. The optional [...] denotes the option of `color` or `xcolor`'s `\color` command. For spot colors, `l3color` module is well supported in PDF and DVI mode. Package `colorspace` is also supported in PDF mode, but could conflict with `luamplib`'s special features such as uncolored tiling pattern when `\DocumentMetadata`, i.e. PDF management code, is not loaded.

N.B. Formerly, only the first object would have been colored as intended among multiple graphical objects in a `METAPOST` image, because `\mpcolor` always produced `withprescript` command internally. Since v2.38.1, now that `\mpcolor` returns a `METAPOST` color expression if

possible, users can issue the sentence as follows without worrying about the location of the color command:

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



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

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

1.1.14 `\mpfig ... \endmpfig`

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

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

and the starred version is roughly the same as follows:

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

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

```
\everymplib[@mpfig]{ drawoptions(withpen pencircle scaled 1 withcolor 2/3red); }
\mpfig* input boxes \endmpfig
\mpfig
  circleit.a(btex Box 1 etex); drawboxed(a);
\endmpfig
```



Users can change the instance name (default value: `@mpfig`) by redefining `\mpfiginstancename`, after which a new `mplib` instance will start and code inheritance too will begin anew. `\let \mpfiginstancename\empty` will prevent code inheritance if `\mplibcodeinherit` is not true.

1.1.15 About cache files

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

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

where `⟨filename⟩` is a filename without `.mp` extension. Note that `.mp` files under `$TEXMFMAIN/metapost/base` and `$TEXMFMAIN/metapost/context/base` are already registered by default.

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

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

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

1.1.16 About figure box metric

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

1.1.17 luamplib.cfg

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

1.1.18 Tagged PDF

When `tagpdf` package is loaded and activated, `mplibcode` environment accepts additional options for tagged PDF. The code related to this functionality is currently in experimental stage, not guaranteeing backward compatibility. Available optional keys are similar to those of the `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 T_EX's regular math formula tagging, for which the `text` key is needed.

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

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

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

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

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

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

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

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

1.2 METAPOST

1.2.1 T_EX dimen and color

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

1.2.2 More on T_EX color

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

the `withcolor` command.⁹ For instance:

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

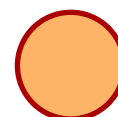
But the result may vary in its color model (gray/rgb/cmyk) according to the given $\TeX$ color. Therefore the example shown above would raise a `METAPOST error: cmykcolor col; should have been declared`. By contrast, `mplibrbgtexcolor` $\langle string \rangle$ always returns rgb-model expressions.

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

1.2.3 Fill and stroke colors

Unlike the `withcolor` command, users can specify one color for filling and another color for stroking using the macro `withmplibcolors` at the end of a sentence. The syntax is `withmplibcolors` $(\langle fill\ color\ expr \rangle, \langle stroke\ color\ expr \rangle)$. When the argument is in string type, it is regarded as the color expression of $\TeX$ side. A simple example (see also the example at § 1.2.8):

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

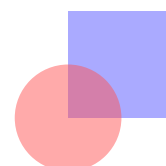


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 Transparency

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

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

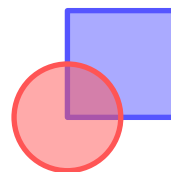


⁹Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color.

1.2.5 Fill and stroke transparency

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

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



1.2.6 Text outline as a text image

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

```
draw mplibgraphicstext "$\sqrt{2+x}$"
  rotated 10 scaled 3
  fakebold 2.5                % fontspec option
  fillcolor "red!50"           % color expression
  drawcolor 2/3 red             % or strokecolor 2/3 red
  ;
```



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

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

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

¹¹But this limitation is now lifted by the introduction of `withshadingmethod`. See below § 1.2.12.

1.2.7 Glyph outline

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

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

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

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

1.2.8 Drawing a glyph outline

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

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

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

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

¹²The bug seems to be fixed in `font-cff.lmt` contained in Con $\TeX$ t mkxl, but current `luaotfload` is based on `font-cff.lua` from Con $\TeX$ t mkiv. As you see, `mplibglyph` operator requires `luaotfload` package loaded, which however is done automatically by $\LaTeX$ format.

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

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`, `mplibgraphictext` (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 filled *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.

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

¹⁴See 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 and transparency group.

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

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

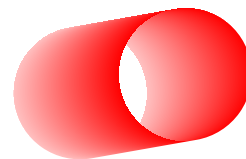
withshadingextend ($\langle boolean \rangle$, $\langle boolean \rangle$) Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

```
\mpfig
path p[];
p1 = fullcircle scaled 50;
p2 = fullcircle scaled 50 shifted 40 right;
```

```

fill (subpath (2,6) of p1 -- subpath (-2,2) of p2 -- cycle) rotated 10
  withshadingmethod "circular"
  withshadingcenters (center p1, center p2 rotated 10)
  withshadingradius (25, 25)
  withshadingcolors (3/4[red,white], red)
  withshadingextend (false, false) ;
\endmpfig

```



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

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

withfadecenter is a synonym of **withfadevector**.

withfaderadius ($\langle \text{numeric} \rangle$, $\langle \text{numeric} \rangle$) sets the radii of starting and ending circles. This is no-op in the linear mode. Default value is (0, abs(center p - urcorner p)), meaning that fading starts from the center and ends at the four corners of the bounding box.

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

withfadefraction $\langle \text{numeric} \rangle$ Fractional number of each fading step. Only meaningful within **withfadestep**.

withfadeextend ($\langle \text{boolean} \rangle$, $\langle \text{boolean} \rangle$) specifies whether to extend the fading beyond the starting and ending points or circles, default value being (true, true).

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

An example:

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



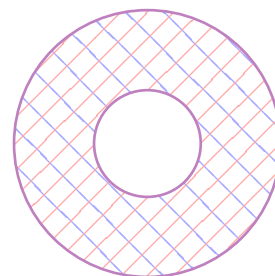
1.2.14 Tiling pattern

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

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;
```



¹⁵withpattern 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, $\langle cyclic path \rangle$ withmppattern $\langle string \rangle$ 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 or string</i>	llx, lly, urx, ury values*
matrix	<i>table or string</i>	xx, yx, xy, yy values* or MP transform code
resources	<i>string</i>	PDF resources if needed
colored or coloured	<i>boolean</i>	false for uncolored pattern. default: true

* in string type, numbers are separated by spaces

```

)
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;

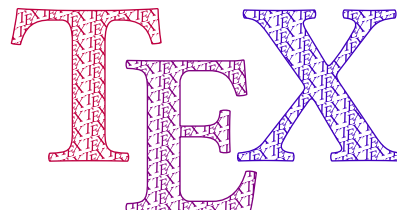
```

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

```

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}

```



1.2.15 Form XObject 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: `<picture> | <path> asgroup <string>`, 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 $\TeX$ code or in other `METAPOST` code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide $\TeX$ and `METAPOST` macros as follows:

withgroupname $\langle string \rangle$ associates 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{ $\langle name \rangle$ } is a $\TeX$ 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 $\langle string \rangle$ is a `METAPOST` command which will add an XObject of the name to the currentpicture. Contrary to the $\TeX$ command just mentioned, the position of the XObject is the same as the original XObject.

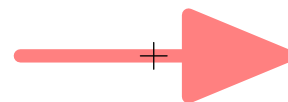
withgroupbbox ($\langle pair \rangle$, $\langle pair \rangle$) sets the bounding box of the XObject, default value being (llcorner p, urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence ‘`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 $\TeX$ and `METAPOST` commands:

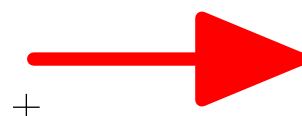
```
\mpfig
picture pic;
pic = image(drawarrow (left--right) scaled 5 withcolor red) scaled 10 ;
draw pic
  asgroup "off"
  withtransparency (1, 1/2) ;
\endmpfig
```



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



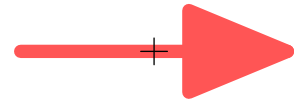
```
\noindent
\clap{\vrule width 10bp height .25bp depth .25bp}%
\clap{\vrule width .5bp height 5bp depth 5bp}%
\usemplibgroup{mygroup}
```



```

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

```



Also note that normally the 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:

```

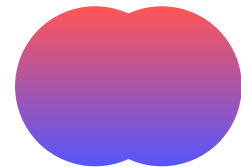
\mpfig*
  picture pic[];
  pic1 = image(
    fill fullcircle scaled 60 withoutcolor;
    fill fullcircle scaled 60 shifted 25right withoutcolor;
  ) ;
  pic2 = image(
    draw pic1
      withshadingmethod "linear"
      withshadingvector (2, 1)
      withshadingcolors (red, blue)
  ) ;
\endmpfig

```

```

\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

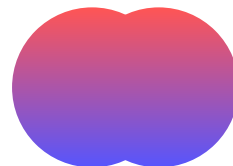
```



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
```



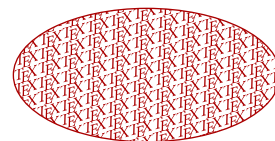
1.2.16 Form XObject from T_EX side

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}
```



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

```
\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.

Option asgroup="wrapped" can be regarded as a shortcut of double \mplibgroup command. The content will be wrapped up in an ordinary form XObject 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 form XObject, users can specify the color space of a transparency group. For instance, the mplibgroup will be painted in grayscale model when colorspace="/DeviceGray" is given to an *isolated* transparency group.¹⁸

As shown, you can reuse the mplibgroup using the TeX command \usemplibgroup or the METAPOST command usemplibgroup. The behavior of these commands is the same as that described [above](#) at § 1.2.15, excepting that the mplibgroup made by TeX 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.

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

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:

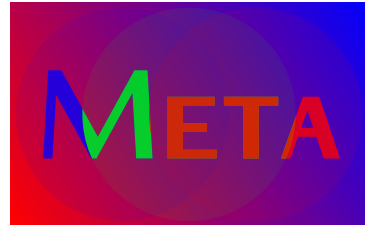
withmaskingbgcolor $\langle$ *color expr* $\rangle$ sets the background color of the masking group. 0 denotes full transparency (masking invisibly); 1, full color.²⁰

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:

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

withtrianglepatchinit (*<path>* | *<string>*, *<color>*, *<color>*, *<color>*) 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 (*<number>*, *<pair>* | *<string>*, *<color>*) The new triangle attached to the previous one. This optional macro requires **withtrianglepatchinit** specified beforehand.

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.

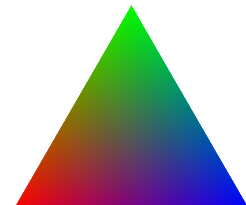
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (*x* and *y* coordinates).

The third argument is the color for the new vertex.

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

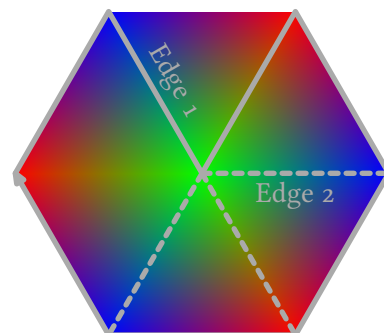
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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 ($\langle pair \rangle$ | $\langle string \rangle$, $\langle color \rangle$, ...) A list of vertices and colors.

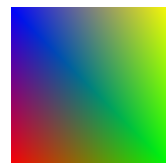
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

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.

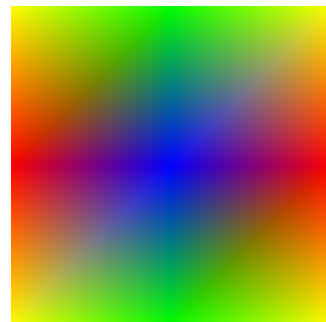
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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 $\langle path \rangle$ | $\langle textual picture \rangle$ `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 (*⟨number⟩*, *⟨path⟩* | *⟨string⟩*, *⟨color⟩*, *⟨color⟩*) The new patch attached to the previous one. This optional macro requires withcoonspatchinit specified beforehand.

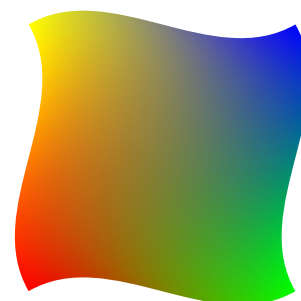
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

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
```

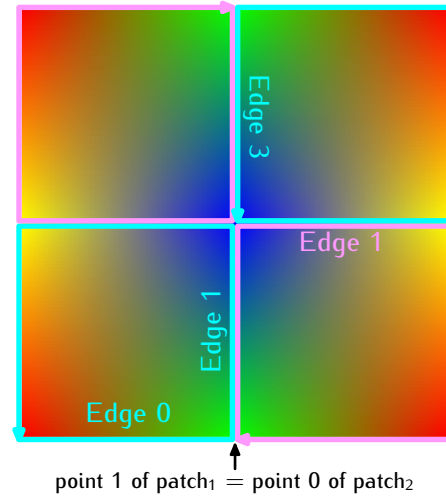


```
\mpfig
u := 80 ;
path p;
p = unitsquare scaled u;
def fsquare (expr n) =
  ( point n of p --
    point n+1 of p --
    point n+2 of p --
    point n+3 of p --
    cycle )
enddef;
```

```

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

```

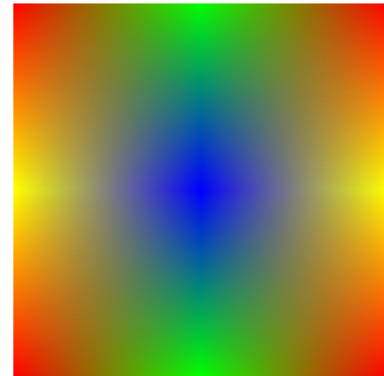


N.B. Be aware that currently Mac OS Preview exposes its some bug in rendering the figure just above, especially when the edge flag is 3. In order to circumvent the Preview's bug, you can use withcoonspatchinit multiple times:

```

\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.

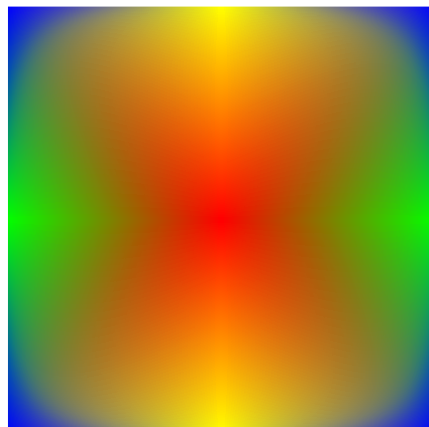
The only difference to `withcoonspatchinit` macro of Coons patch shading is the second argument inserted for specifying four inner control points. It shall be a path that has four points, or a string composed of eight numerics separated by spaces (x and y coordinates of each point). If the first argument 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.

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> (<i>(string)</i>)	<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> (<i>(string)</i>)	<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

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)
```

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 ‘MP11x’ and ‘MP11y’ 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.

²¹In DVI mode, T_EX macro ‘`mplibpatternname`’ should be set as `<pattern name>` 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 `<group name>` before preparing the box, if shading pattern (i.e. shading on picture) is used in the `mplibgroup`.

2 Implementation

2.1 Lua module

```
1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.42.5",
5   date      = "2026/08/07",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8
```

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

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

```

42 termorlog("log", select("#",...) > 1 and format(...) or ...)
43 end
44 local function err (...)
45   termorlog("error", select("#",...) > 1 and format(...) or ...)
46 end
47
48 luamplib.showlog = luamplib.showlog or false
49

```

Provide a few “shortcuts” expected by the code.

```

50 local tableconcat = table.concat
51 local tableinsert = table.insert
52 local tableunpack = table.unpack
53 local texsprint   = tex.sprint
54 local texgettoks  = tex.gettoks
55 local texgetbox    = tex.getbox
56 local texruntoks   = tex.runtoks
57 if not texruntoks then
58   err("Your LuaTeX version is too old. Please upgrade it to the latest")
59 end
60 local is_defined = token.is_defined
61 local get_macro  = token.get_macro
62 local mplib = require ('mplib')
63 local kpse = require ('kpse')
64 local lfs = require ('lfs')
65 local lfsattributes = lfs.attributes
66 local lfsisdir      = lfs.isdir
67 local lfsmkdir      = lfs.mkdir
68 local lfstouch      = lfs.touch
69 local 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.mkdir or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\w/]+)") do

```

```

88   full = full .. sub
89   lfsmkdir(full)
90 end
91 end
92

```

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

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

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

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

```

```

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

```

Pattern formats to replace btex and verbatimtex ... 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 verbatimtex_etex = name_b.."verbatimtex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
151

```

Function luamplib.finder

```

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

```

format.mp is much complicated, so specially treated.

```

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

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

For the main function: process

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

```

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

```

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

```

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

```

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

```

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

```

```

265         termorlog("term", first)
266         termorlog("log", log, "Warning")
267     else
268         warn(log)
269     end
270     if result.status > 1 then
271         err(e or "see above messages")
272     end
273 elseif prevlog then
274     log = prevlog..log

```

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

299 local preamble = tableconcat{
300     format(luamplib.preambles.preamble, replacesuffix(name,"mp")),
301     luamplib.preambles.mplibcode,
302     luamplib.legacyverbatim and luamplib.preambles.legacyverbatim or "",
303     luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
304 }

```

```

305     local result, log
306     if not mpx then
307         result = { status = 99, error = "out of memory"}
308     else
309         result = mpx:execute(preamble)
310     end
311     log = reporterror(result)
312     return mpx, result, log
313 end

```

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

```

314 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() texsprint(cat or catlatex, str) end)
364 end

```

For conversion of sp to bp.

```

365 local factor = 65536*(7227/7200)
366

```

Prepare texttext box number containers, locals and globals. localid can be any number. They are local anyway. The number will be reset at the start of a new code chunk. Global boxes will use \newbox command in tex.runtoks process. This is the same when codeinherit is true. Boxes in instances with name will also be global, so that their tex boxes can be shared among instances of the same name.

```

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 texttext_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"^%d" 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\mplibtmptoks\expandafter{\current@color}}}}]],
441   l3color = is_defined "__color_export_format_raw:nnN" and
442     [[\color_export:nnN{s}{raw}\l_tmpa_tl\mplibtmptoks\expandafter{\l_tmpa_tl}]]
443   or [[{\setbox0\hbox{{\color_select:n\s\global\mplibtmptoks\expanded{{\current@color}}}}}}]]
444 }
445 function process_color (str)
446   if str then
447     if not str:find("%b{") then
448       str = format("{%s}",str)
449     end
450     local myfmt = mplibcolorfmt[colfmt]
451     if colfmt == "l3color" and is_defined "color" then
452       if str:find("%b[") or is_xcolor(str:match"{(.+)}") then
453         myfmt = mplibcolorfmt.xcolor
454       end
455     end
456     run_tex_code(myfmt:format(str), ccexplat or catat11)
457     local t = texgettoks "mplibtmptoks"
458     if not pdfmode then
459       ---[[ to be removed
460       if t:find"^ color%d" then
461         local tt = t:explode()
462         t = ("%s cs %s scn /%s CS %s SCN"):format(tt[1], tt[2], tt[1], tt[2])
463       --]]
464       elseif t:find"^hsb" then
465         local spec = t:gsub("^hsb ", ""):gsub(" ", ",")
466         run_tex_code{"\\convertcolorspec{hsb}{", spec, "}{rgb}\\mplibtmpa"}
467         t = format("rgb %s", get_macro "mplibtmpa":gsub(","," "))
468       elseif not t:find"%d" then -- named color
469         run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
470         t = format("%s %s", get_macro "mplibtmpa", get_macro "mplibtmpb":gsub(","," "))
471       end
472       local name, value = t:match("(%a+) (.+)")
473       if name and value then
474         local op = name == "rgb" and "rg" or name == "cmyk" and "k" or name == "gray" and "g"
475         if not op then err "unknown color model" end
476         t = ("%s %s %s %s"):format(value, op, value, op:upper())
477       elseif t:find" cs " then -- spot color raw export
478         name = t:match("^/(.-) cs ")
479         texsprint(ccexplat, {
480           "\\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
481             name, "}{\\pdf_object_ref:n{", name, "}}"
482         })
483       end

```

```

484     elseif is_defined"ver@colorspace.sty" and t:find"^/&" then
485         local a,b,c,d = t:match"^(.- cs) (.- CS) (.- scn?) (.- SCN?)$"
486         if a and b and c and d then
487             t = tableconcat({ a, c, b, d }, " ")
488         end
489     end
490     return format('1 withprescript "mpliboverridecolor=%s"', t)
491 end
492 return ""
493 end
494 function process_mplibcolor(str)
495     local res = process_color(str)
496     if res:find" cs " then return res end
497     res = colorsplit(res:match'"mpliboverridecolor=(.)"')
498     return format("(%s)", tableconcat(res, ","))
499 end
500 end
501
    for \mpdim or mplibdimen
502 local function process_dimen (str)
503     if str then
504         str = str:gsub("{(.+)}", "%1")
505         run_tex_code(format([[ \mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
506         return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
507     end
508     return ""
509 end
510

```

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

```

511 local function process_verbatimtex_text (str)
512     if str then
513         run_tex_code(str)
514     end
515     return ""
516 end
517

```

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.

```

518 local tex_code_pre_mplib = {}
519 luamplib.figid = 1
520 luamplib.in_the_fig = false
521 local function process_verbatimtex_prefig (str)
522     if str then
523         tex_code_pre_mplib[luamplib.figid] = str
524     end
525     return ""
526 end

```

```

527 local function process_verbatimtex_infig (str)
528   if str then
529     return format('special "postmplibverbtx=%s";', str)
530   end
531   return ""
532 end
533

```

For *metafun* format. see issue #79.

```

534 mp = mp or {}
535 local mp = mp
536 mp.mf_path_reset = mp.mf_path_reset or function() end
537 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
538 mp.report = mp.report or info

```

metafun 2021-03-09 changes crashes luamplib.

```

539 catcodes = catcodes or {}
540 local catcodes = catcodes
541 catcodes.numbers = catcodes.numbers or {}
542 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
543 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
544 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
545 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
546 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
547 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
548 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
549

```

Now luamplib.runscript

```

550 do
551   local runscript_funcs = {
552     luamplibtext    = process_tex_text,
553     luamplibcolor   = process_mplibcolor,
554     luamplibdimen   = process_dimen,
555     luamplibprefig  = process_verbatimtex_prefig,
556     luamplibinfig   = process_verbatimtex_infig,
557     luamplibverbtx  = process_verbatimtex_text,
558   }

```

A function from ConT_EXt general.

```

559 local function mpprint(buffer,...)
560   for i=1,select("#",...) do
561     local value = select(i,...)
562     if value ~= nil then
563       local t = type(value)
564       if t == "number" then
565         buffer[#buffer+1] = format("%.16f",value)
566       elseif t == "string" then
567         buffer[#buffer+1] = value
568       elseif t == "table" then
569         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
570       else -- boolean or whatever

```

```

571         buffer[#buffer+1] = tostring(value)
572     end
573 end
574 end
575 end
576 function luamplib.runscript (code)
577     local id, str = code:match("(.-){(.*)}")
578     if id and str then
579         local f = runscript_funcs[id]
580         if f then
581             local t = f(str)
582             if t then return t end
583         end
584     end
585     local f = loadstring(code)
586     if type(f) == "function" then
587         local buffer = {}
588         function mp.print(...)
589             mpprint(buffer,...)
590         end
591         local res = {f()}
592         buffer = tableconcat(buffer)
593         if buffer and buffer ~= "" then
594             return buffer
595         end
596         buffer = {}
597         mpprint(buffer, tableunpack(res))
598         return tableconcat(buffer)
599     end
600     return ""
601 end
602 end
603
604     luamplib.maketext
605     luamplib.legacyverbatimimtex = true
606 do

```

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

```

606     local function protecttexcontents (str)
607         return str:gsub("\\%", "\\0Percent\0")
608             :gsub("%%.-\n", "")
609             :gsub("%%.-$", "")
610             :gsub("%zPercent%z", "\\0Percent\0")
611             :gsub("%r.-$", "")
612             :gsub("%s+", " ")
613     end
614     function luamplib.maketext (str, what)
615         if str and str ~= "" then
616             str = protecttexcontents(str)

```

```

617   if what == 1 then
618       if not str:find("\\documentclass"..name_e) and
619           not str:find("\\begin%s*{document}") and
620           not str:find("\\documentstyle"..name_e) and
621           not str:find("\\usepackage"..name_e) then
622           if luamplib.legacyverbatim then
623               if luamplib.in_the_fig then
624                   return process_verbatim_infig(str)
625               else
626                   return process_verbatim_prefig(str)
627               end
628           else
629               return process_verbatim_text(str)
630           end
631       end
632   else
633       return process_tex_text(str, true) -- bool is for 'char13'
634   end
635 end
636 return ""
637 end
638 end
639

```

luamplib's METAPOST color operators

```

640 local function get_spc_name_obj (spot)
641   if is_defined"spc@csall" then
642       for v in get_macro"spc@csall":gmatch"{{(.-)}}" do
643           local n, r = get_macro("spc@ir@"..v):match"^(.-) (%d+ 0 R)"
644           if spot == n then
645               return v, r
646           end
647       end
648   else
649       err"only l3color or colorspace is supported for spot color"
650   end
651 end
652 do
653   local function cmyk2rgb (t)
654       return {
655           1 - math.min(1, t[1]+t[4]),
656           1 - math.min(1, t[2]+t[4]),
657           1 - math.min(1, t[3]+t[4]),
658       }
659   end
660   luamplib.gettexcolor = function (str, rgb)
661       local res = process_color(str):match'"mpliboverridecolor=(.+)"'
662       if res:find" cs " then
663           if not rgb then
664               warn("%s is a spot color. Forced to CMYK", str)

```

```

665     end
666     if not is_xcolor(str) then
667         run_tex_code({
668             "\\color_export:nnN{" ,
669             str,
670             "}{",
671             rgb and "space-sep-rgb" or "space-sep-cmyk",
672             "\\mplib@tempa",
673         },ccexplat)
674         return get_macro"mplib@tempa":explode()
675     else
676         local name, value = res:match"^(.-) cs (.-) sc"
677         local t = get_macro("spc@ascmyk@"..get_spc_name_obj(name)):explode", " -- mixed not work
678         for i = 1, #t do
679             t[i] = t[i] * value
680         end
681         return rgb and cmyk2rgb(t) or t
682     end
683 end
684 local t = colorsplit(res)
685 if #t == 3 or not rgb then return t end
686 if #t == 4 then return cmyk2rgb(t) end
687 return { t[1], t[1], t[1] }
688 end
689 end
690 luamplib.shadecolor = function (str)
691     local res = process_color(str):match'"mpliboverridecolor=(.+)"'
692     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}

```

693 local name, value, obj

```

694   if not is_xcolor(str) then
695     run_tex_code({ "\\color_export:nnN{", str, "{\\backend}\\mplib@tempa" }, ccexplat)
696     name, value = get_macro'mplib@tempa':match'{{(.-)}}{(.-)}'
697     local t = res:explode()
698     local refname = t[1]:sub(2,-1)
699     if pdfmode then
700       obj = format("%s 0 R", ltx.pdf.object_id(refname))
701     else
702       run_tex_code({ "\\mplibtmp toks\\expanded{{\\pdf_object_ref:n{", refname, "}}}" }, ccexplat)
703       obj = texgettoks"mplibtmp toks"
704     end
705   else -- colorspace: mixing is allowed only in preamble
706     name, value = res:match"^(.-) cs (.-) sc"
707     name, obj = get_spc_name_obj(name)
708   end
709   return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
710 end
711 return format('(%s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),","), str)
712 end
713

```

luamplib.fillandstrokecolor

```

714 do
715   local function graphictextcolor (col, filldraw)
716     if col:find"^[%d%.:]+$" then
717       col = col:explode":"
718       for i=1,#col do
719         col[i] = format("%.3f", col[i])
720       end
721       local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
722       col[#col+1] = filldraw == "fill" and op or op:upper()
723       return tableconcat(col," ")
724     end
725     col = process_color(col):match'"mpliboverridecolor=(.+)"'
726     local t = col:explode()
727     local b = filldraw == "fill" and 1 or #t/2+1
728     local e = b == 1 and #t/2 or #t
729     return tableconcat(t," ", b, e)
730   end
731   function luamplib.fillandstrokecolor (fill, stroke)
732     fill = graphictextcolor(fill, "fill")
733     stroke = graphictextcolor(stroke, "stroke")
734     return format('withprescript "mpliboverridecolor=%s %s"', fill, stroke)
735   end
736 end
737

```

Remove trailing zeros for smaller PDF

```

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

```

740

common function for mplibgraphicstext and mpliboutlinetext

```
741 local function getrulemetric (box, curr, bp)
742   local running = -1073741824
743   local wd,ht,dp = curr.width, curr.height, curr.depth
744   wd = wd == running and box.width or wd
745   ht = ht == running and box.height or ht
746   dp = dp == running and box.depth or dp
747   if bp then
748     return wd/factor, ht/factor, dp/factor
749   end
750   return wd, ht, dp
751 end
752
```

luamplib's mplibgraphicstext operator

```
753 do
754   if not math.round then
755     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
756   end
757   local emboldenfonts = { }
758   local function roundupwidth (f, fb)
759     local wd = math.round(f.size * fb / factor * 10)
760     if wd == 0 and fb ~= 0 then
761       wd = 1
762     end
763     emboldenfonts.width = wd
764     return wd
765   end
766   local function getemboldenwidth (curr, fakebold)
767     local width = emboldenfonts.width
768     if not width then
769       local f
770       local function getglyph(n)
771         while n do
772           if n.head then
773             getglyph(n.head)
774           elseif n.font and n.font > 0 then
775             f = n.font; break
776           end
777           n = node.getnext(n)
778         end
779       end
780       getglyph(curr)
781       width = roundupwidth(font.getcopy(f or font.current()), fakebold)
782     end
783     return width
784   end
785   local function getrulewhatsit (line, wd, ht, dp)
```

```

786 line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
787 line = line == 0 and "" or ("%f w"):format(line)
788 local pl
789 local fmt = "q %s %f %f %f %f re B Q"
790 if pdfmode then
791   pl = node.new("whatsit", "pdf_literal")
792   pl.mode = 0
793 else
794   fmt = "pdf:content " .. fmt
795   pl = node.new("whatsit", "special")
796 end
797 pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
798 local ss = node.new"glue"
799 node.setglue(ss, 0, 65536, 65536, 2, 2)
800 pl.next = ss
801 return pl
802 end

```

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

```

803 local tag_update_attrs
804 if is_defined"ver@tagpdf.sty" then
805   tag_update_attrs = function (n, curr)
806     while n do
807       n.attr = curr.attr
808       if n.head then
809         tag_update_attrs(n.head, curr)
810       end
811       n = node.getnext(n)
812     end
813   end
814 else
815   tag_update_attrs = function() end
816 end
817 local function embolden (box, curr, fakebold)
818   local head = curr
819   while curr do
820     if curr.head then
821       curr.head = embolden(curr, curr.head, fakebold)
822     elseif curr.replace then
823       curr.replace = embolden(box, curr.replace, fakebold)
824     elseif curr.leader then
825       if curr.leader.head then
826         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
827       elseif curr.leader.id == node.id"rule" then
828         local glue = node.effective_glue(curr, box)
829         local line = getemboldenwidth(curr, fakebold)
830         local wd, ht, dp = getrulemetric(box, curr.leader)
831         if box.id == node.id"hlist" then
832           wd = glue

```

```

833     else
834         ht, dp = 0, glue
835     end
836     local pl = getrulewhatsit(line, wd, ht, dp)
837     local pack = box.id == node.id"hlist" and node.hpack or node.vpack
838     local list = pack(pl, glue, "exactly")
839     tag_update_attrs(list, curr)
840     head = node.insert_after(head, curr, list)
841     head, curr = node.remove(head, curr)
842 end
843 elseif curr.id == node.id"rule" and curr.subtype == 0 then
844     local line = getemboldenwidth(curr, fakebold)
845     local wd, ht, dp = getrulemetric(box, curr)
846     if box.id == node.id"vlist" then
847         ht, dp = 0, ht+dp
848     end
849     local pl = getrulewhatsit(line, wd, ht, dp)
850     local list
851     if box.id == node.id"hlist" then
852         list = node.hpack(pl, wd, "exactly")
853     else
854         list = node.vpack(pl, ht+dp, "exactly")
855     end
856     tag_update_attrs(list, curr)
857     head = node.insert_after(head, curr, list)
858     head, curr = node.remove(head, curr)
859 elseif curr.id == node.id"glyph" and curr.font > 0 then
860     local f = curr.font
861     local key = format("%s:%s", f, fakebold)
862     local i = emboldenfonts[key]
863     if not i then
864         local ft = font.getfont(f) or font.getcopy(f)
865         local width = roundupwidth(ft, fakebold)
866         if ft.format == "opentype" or ft.format == "truetype" then
867             local name = ft.name.gsub("'", ''):gsub('$', '')
868             local t = name.gsub("^file:", ""):gsub("^name:", ""):gsub("^kpse:", ""):gsub("^my:", "")
869             name = format('%s%sembolden=%s;', name, t:find":" and ";" or ":", fakebold)
870             _, i = fonts.constructors.readanddefine(name, ft.size)
871         elseif pdfmode then
872             local ft = table.copy(ft)
873             ft.mode, ft.width = 2, width
874             i = font.define(ft)
875         else
876             goto skip_type1
877         end
878         emboldenfonts[key] = i
879     end
880     curr.font = i
881 end

```

```

882     ::skip_type1::
883     curr = node.getnext(curr)
884 end
885 return head
886 end
887 luamplib.graphicstext = function (text, fakebold, fc, dc)
888     local fmt = process_tex_text(text):sub(1,-2)
889     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
890     emboldenfonts.width = nil
891     local box = texgetbox(id)
892     box.head = embolden(box, box.head, fakebold)
893     local colors = luamplib.fillandstrokecolor(fc, dc)
894     return format('%s %s)', fmt, colors)
895 end
896 end
897

```

luamplib's mplibglyph operator

```

898 do
899     local function mperr (str)
900         return format("hide(errmessage %q)", str)
901     end
902     local function getangle (a,b,c)
903         local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
904         if r > 180 then
905             r = r - 360
906         elseif r < -180 then
907             r = r + 360
908         end
909         return r
910     end
911     local function turning (t)
912         local r, n = 0, #t
913         for i=1,2 do
914             tableinsert(t, t[i])
915         end
916         for i=1,n do
917             r = r + getangle(t[i], t[i+1], t[i+2])
918         end
919         return r/360
920     end
921     local function glyphimage(t, fmt)
922         local q, p, r, towarn = {},{}
923         local function closepath(dots)
924             tableinsert(p, format("%scycle", dots or "--"))
925             tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
926         end
927         for i,v in ipairs(t) do
928             local cmd = v[#v]

```

```

929     local nt = t[i+1]
930     local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
931     if cmd == "m" then
932         if final then towarn = true end
933         p = {format('(%s,%s)',v[1],v[2])}
934         r = {{x=v[1],y=v[2]}}
935     else
936         if cmd == "l" then
937             local pt = t[i-1]
938             if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
939                 else
940                     tableinsert(p, format('--(%s,%s)',v[1],v[2]))
941                     tableinsert(r, {x=v[1],y=v[2]})
942                 end
943             if final then closepath() end
944         elseif cmd == "c" then
945             tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
946             if final and r[1].x == v[5] and r[1].y == v[6] then
947                 closepath ".."
948             else
949                 tableinsert(p, format('..(%s,%s)',v[5],v[6]))
950                 tableinsert(r, {x=v[5],y=v[6]})
951                 if final then closepath() end
952             end
953         elseif cmd == "path" or cmd == "move" then
954             else
955                 return mperr"unknown operator"
956             end
957         end
958     end
959     r = { }
960     if fmt == "opentype" then
961         for _,v in ipairs(q[1]) do
962             tableinsert(r, format('addto currentpicture contour %s;',v))
963         end
964         for _,v in ipairs(q[2]) do
965             tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
966         end
967     else
968         for _,v in ipairs(q[2]) do
969             tableinsert(r, format('addto currentpicture contour %s;',v))
970         end
971         for _,v in ipairs(q[1]) do
972             tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
973         end
974     end
975     return format('image(%s)', tableconcat(r)), towarn
976 end
977 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end

```

```

978 function luampplib.glyph (f, c)
979   local filename, subfont, instance, kind, shapedata
980   local fid = tonumber(f) or font.id(f)
981   if fid > 0 then
982     local fontdata = font.getfont(fid) or font.getcopy(fid)
983     filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
984     instance = fontdata.specification and fontdata.specification.instance
985     or fontdata.shared and fontdata.shared.features.axis
986     filename = filename and filename:gsub("^harfloaded:", "")
987   else
988     local name
989     f = f:match"^%s*(.)%s*$"
990     name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
991     if not name then
992       name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
993     end
994     if not name then
995       name, subfont = f:match"(.+)%((%d+)%)$" -- Times.ttc(2)
996     end
997     name = name or f
998     subfont = (subfont or 0)+1
999     instance = instance and instance:lower()
1000    for _,ftype in ipairs{"opentype", "truetype"} do
1001      filename = kpse.find_file(name, ftype.." fonts")
1002      if filename then
1003        kind = ftype; break
1004      end
1005    end
1006  end
1007  if kind ~= "opentype" and kind ~= "truetype" then
1008    f = fid and fid > 0 and tex.fontname(fid) or f
1009    if kpse.find_file(f, "tfm") then
1010      return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
1011    else
1012      filename = kpse.find_file(f, "type1 fonts")
1013      if filename then
1014        kind = "type1" -- there's bug in processing cmr family
1015      else
1016        return mperr"font not found"
1017      end
1018    end
1019  end
1020  local time = lfsattributes(filename,"modification")

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

1021  local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1022  local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))

```

```

1023     local newname = format("%s/%s.lua", cachedir or outputdir(), h)
1024     local newtime = lfsattributes(newname,"modification") or 0
1025     if time == newtime then
1026         shapedata = require(newname)
1027     end
1028     if not shapedata then
1029         if fonts then
1030             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1031             shapedata = handler.readers.loadshapes(filename,subfont,instance)
1032         end
1033         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1034         table.tofile(newname, shapedata, "return")
1035         lfstouch(newname, time, time)
1036     end
1037     local gid = tonumber(c)
1038     if not gid then
1039         local uni = utf8.codepoint(c)
1040         for i,v in pairs(shapedata.glyphs) do
1041             if c == v.name or uni == v.unicode then
1042                 gid = i; break
1043             end
1044         end
1045     end
1046     if not gid then return mperr"cannot get GID (glyph id)" end
1047     local fac = 1000 / (shapedata.units or 1000)
1048     local t = shapedata.glyphs[gid]; t = t and t.segments
1049     if not t then return "image()" end
1050     for i,v in ipairs(t) do
1051         if type(v) == "table" then
1052             for ii,vv in ipairs(v) do
1053                 if type(vv) == "number" then
1054                     t[i][ii] = format("%.0f", vv * fac)
1055                 end
1056             end
1057         end
1058     end
1059     local result, towarn = glyphimage(t, shapedata.format or kind)
1060     if towarn then
1061         warn("mplibglyph %s not working properly. Use glyph instead", f)
1062     end
1063     return result
1064 end
1065 end
1066

```

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

```

1067 do
1068     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1069         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"

```

```

1070 local outline_horz, outline_vert
1071 function outline_vert (res, box, curr, xshift, yshift)
1072     local b2u = box.dir == "LTL"
1073     local dy = (b2u and -box.depth or box.height)/factor
1074     local ody = dy
1075     while curr do
1076         if curr.id == node.id"rule" then
1077             local wd, ht, dp = getrulemetric(box, curr, true)
1078             local hd = ht + dp
1079             if hd ~= 0 then
1080                 dy = dy + (b2u and dp or -ht)
1081                 if wd ~= 0 and curr.subtype == 0 then
1082                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1083                 end
1084                 dy = dy + (b2u and ht or -dp)
1085             end
1086         elseif curr.id == node.id"glue" then
1087             local vwidth = node.effective_glue(curr,box)/factor
1088             if curr.leader then
1089                 local curr, kind = curr.leader, curr.subtype
1090                 if curr.id == node.id"rule" then
1091                     local wd = getrulemetric(box, curr, true)
1092                     if wd ~= 0 then
1093                         local hd = vwidth
1094                         local dy = dy + (b2u and 0 or -hd)
1095                         if hd ~= 0 and curr.subtype == 0 then
1096                             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1097                         end
1098                     end
1099                 elseif curr.head then
1100                     local hd = (curr.height + curr.depth)/factor
1101                     if hd <= vwidth then
1102                         local dy, n, iy = dy, 0, 0
1103                         if kind == 100 or kind == 103 then -- todo: gleaders
1104                             local ady = abs(ody - dy)
1105                             local ndy = math.ceil(ady / hd) * hd
1106                             local diff = ndy - ady
1107                             n = math.floor((vwidth-diff) / hd)
1108                             dy = dy + (b2u and diff or -diff)
1109                         else
1110                             n = math.floor(vwidth / hd)
1111                             if kind == 101 then
1112                                 local side = vwidth % hd / 2
1113                                 dy = dy + (b2u and side or -side)
1114                             elseif kind == 102 then
1115                                 iy = vwidth % hd / (n+1)
1116                                 dy = dy + (b2u and iy or -iy)
1117                             end
1118                         end
1119                     end
1120                 end
1121             end
1122         end
1123     end

```

```

1119         dy = dy + (b2u and curr.depth or -curr.height)/factor
1120         hd = b2u and hd or -hd
1121         iy = b2u and iy or -iy
1122         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1123         for i=1,n do
1124             res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1125             dy = dy + hd + iy
1126         end
1127     end
1128 end
1129 end
1130 dy = dy + (b2u and vwidth or -vwidth)
1131 elseif curr.id == node.id"kern" then
1132     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1133 elseif curr.id == node.id"vlist" then
1134     dy = dy + (b2u and curr.depth or -curr.height)/factor
1135     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1136     dy = dy + (b2u and curr.height or -curr.depth)/factor
1137 elseif curr.id == node.id"hlist" then
1138     dy = dy + (b2u and curr.depth or -curr.height)/factor
1139     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1140     dy = dy + (b2u and curr.height or -curr.depth)/factor
1141 end
1142 curr = node.getnext(curr)
1143 end
1144 return res
1145 end
1146 function outline_horz (res, box, curr, xshift, yshift, discwd)
1147     local r2l = box.dir == "TRT"
1148     local dx = r2l and (discwd or box.width/factor) or 0
1149     local dirs = { { dir = r2l, dx = dx } }
1150     while curr do
1151         if curr.id == node.id"dir" then
1152             local sign, dir = curr.dir:match"(...)()"
1153             local level, newdir = curr.level, r2l
1154             if sign == "+" then
1155                 newdir = dir == "TRT"
1156                 if r2l ~= newdir then
1157                     local n = node.getnext(curr)
1158                     while n do
1159                         if n.id == node.id"dir" and n.level+1 == level then break end
1160                         n = node.getnext(n)
1161                     end
1162                     n = n or node.tail(curr)
1163                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1164                 end
1165                 dirs[level] = { dir = r2l, dx = dx }
1166             else
1167                 local level = level + 1

```

```

1168     newdir = dirs[level].dir
1169     if r2l ~= newdir then
1170         dx = dirs[level].dx
1171     end
1172     end
1173     r2l = newdir
1174 elseif curr.char and curr.font and curr.font > 0 then
1175     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1176     local gid = ft.characters[curr.char].index or curr.char
1177     local scale = ft.size / factor / 1000
1178     local slant = (ft.slant or 0)/1000
1179     local extend = (ft.extend or 1000)/1000
1180     local squeeze = (ft.squeeze or 1000)/1000
1181     local expand = 1 + (curr.expansion_factor or 0)/1000000
1182     local xscale, yscale = scale * extend * expand, scale * squeeze
1183     dx = dx - (r2l and curr.width/factor*expand or 0)
1184     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1185     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1186     local vertical = ""
1187     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1188         if ft.shared.features.vertical then -- luatexko
1189             vertical = "rotated 90"
1190             local data = ft.characters[curr.char] or { }
1191             if ft.hb then
1192                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1193                 local charraise = (ft.luatexko_charraise or 0)/factor
1194                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1195             else
1196                 local cmds = data.commands or { {0,0}, {0,0} }
1197                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1198                 xpos, ypos = xpos + hoff, ypos + voff
1199             end
1200         elseif curr ~= box.head then -- luatexja
1201             vertical = "rotated 90"
1202             local en = ft.parameters.quad/factor/2
1203             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1204         end
1205     end
1206     local image
1207     if ft.format == "opentype" or ft.format == "truetype" then
1208         image = luamplib.glyph(curr.font, gid)
1209     else
1210         local name, scale = ft.name, 1
1211         local vf = font.read_vf(name, ft.size)
1212         if vf and vf.characters[gid] then
1213             local cmds = vf.characters[gid].commands or {}
1214             for _,v in ipairs(cmds) do
1215                 if v[1] == "char" then
1216                     gid = v[2]

```

```

1217         elseif v[1] == "font" and vf.fonts[v[2]] then
1218             name = vf.fonts[v[2]].name
1219             scale = vf.fonts[v[2]].size / ft.size
1220         end
1221     end
1222 end
1223 image = format("glyph %s of %q scaled %f", gid, name, scale)
1224 end
1225 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1226                     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1227 dx = dx + (r2l and 0 or curr.width/factor*expand)
1228 elseif curr.replace then
1229     local width = node.dimensions(curr.replace)/factor
1230     dx = dx - (r2l and width or 0)
1231     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1232     dx = dx + (r2l and 0 or width)
1233 elseif curr.id == node.id"rule" then
1234     local wd, ht, dp = getrulemetric(box, curr, true)
1235     if wd ~= 0 then
1236         local hd = ht + dp
1237         dx = dx - (r2l and wd or 0)
1238         if hd ~= 0 and curr.subtype == 0 then
1239             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1240         end
1241         dx = dx + (r2l and 0 or wd)
1242     end
1243 elseif curr.id == node.id"glue" then
1244     local width = node.effective_glue(curr, box)/factor
1245     dx = dx - (r2l and width or 0)
1246     if curr.leader then
1247         local curr, kind = curr.leader, curr.subtype
1248         if curr.id == node.id"rule" then
1249             local wd, ht, dp = getrulemetric(box, curr, true)
1250             local hd = ht + dp
1251             if hd ~= 0 then
1252                 wd = width
1253                 if wd ~= 0 and curr.subtype == 0 then
1254                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1255                 end
1256             end
1257         end
1258     elseif curr.head then
1259         local wd = curr.width/factor
1260         if wd <= width then
1261             local dx = r2l and dx+width or dx
1262             local n, ix = 0, 0
1263             if kind == 100 or kind == 103 then -- todo: gleaders
1264                 local adx = abs(dx-dirs[1].dx)
1265                 local ndx = math.ceil(adx / wd) * wd
1266                 local diff = ndx - adx

```

```

1266         n = math.floor((width-diff) / wd)
1267         dx = dx + (r2l and -diff-wd or diff)
1268     else
1269         n = math.floor(width / wd)
1270         if kind == 101 then
1271             local side = width % wd / 2
1272             dx = dx + (r2l and -side-wd or side)
1273         elseif kind == 102 then
1274             ix = width % wd / (n+1)
1275             dx = dx + (r2l and -ix-wd or ix)
1276         end
1277     end
1278     wd = r2l and -wd or wd
1279     ix = r2l and -ix or ix
1280     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1281     for i=1,n do
1282         res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1283         dx = dx + wd + ix
1284     end
1285 end
1286 end
1287 end
1288 dx = dx + (r2l and 0 or width)
1289 elseif curr.id == node.id"kern" then
1290     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1291 elseif curr.id == node.id"math" then
1292     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1293 elseif curr.id == node.id"vlist" then
1294     dx = dx - (r2l and curr.width/factor or 0)
1295     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1296     dx = dx + (r2l and 0 or curr.width/factor)
1297 elseif curr.id == node.id"hlist" then
1298     dx = dx - (r2l and curr.width/factor or 0)
1299     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1300     dx = dx + (r2l and 0 or curr.width/factor)
1301 end
1302 curr = node.getnext(curr)
1303 end
1304 return res
1305 end
1306 function luamplib.outlinetext (text)
1307     local fmt = process_tex_text(text)
1308     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1309     local box = texgetbox(id)
1310     local res = outline_horz({ }, box, box.head, 0, 0)
1311     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1312     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1313 end
1314 end

```

```

1315
    lua functions for mplib(uc)substring ... of ...
1316 function luamplib.getunicodegraphemes (s)
1317   local t = { }
1318   local graphemes = require'lua-uni-graphemes'
1319   for _, _, c in graphemes.graphemes(s) do
1320     table.insert(t, c)
1321   end
1322   return t
1323 end
1324 function luamplib.unicodesubstring (s,b,e,grph)
1325   local tt, t, step = { }
1326   if grph then
1327     t = luamplib.getunicodegraphemes(s)
1328   else
1329     t = { }
1330     for _, c in utf8.codes(s) do
1331       table.insert(t, utf8.char(c))
1332     end
1333   end
1334   if b <= e then
1335     b, step = b+1, 1
1336   else
1337     e, step = e+1, -1
1338   end
1339   for i = b, e, step do
1340     table.insert(tt, t[i])
1341   end
1342   s = table.concat(tt):gsub("'", "'&ditto'")
1343   return string.format("%s", s)
1344 end
1345
    METAPOST preambles
1346 luamplib.preambles = {
1347   preamble = [[
1348 boolean mplib ; mplib := true ;
1349 let dump = endinput ;
1350 let normalfontsize = fontsize;
1351 input %s ;
1352 ]],
1353   mplibcode = [[
1354 texscriptmode := 2;
1355 def rawtexttext primary t = runscript("luamplibtext{"&t;"}") enddef;
1356 def mplibcolor primary t = runscript("luamplibcolor{"&t;"}") enddef;
1357 def mplibdimen primary t = runscript("luamplibdimen{"&t;"}") enddef;
1358 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t;"}") enddef;
1359 if known context_mlib:
1360   defaultfont := "cmtt10";

```

```

1361 let infont = normalinfont;
1362 let fontsize = normalfontsize;
1363 vardef thelabel@#(expr p,z) =
1364   if string p :
1365     thelabel@#(p infont defaultfont scaled defaultscale,z)
1366   else :
1367     p shifted (z + labeloffset*mfun_laboff@# -
1368       (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1369       (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1370   fi
1371 enddef;
1372 else:
1373 vardef texttext@# primary t = rawtexttext (t) enddef;
1374 def message expr t =
1375   if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1376 enddef;
1377 def withtransparency (expr a, t) =
1378   withprescript "tr_alternative=" & if numeric a: decimal fi a
1379   withprescript "tr_transparency=" & decimal t
1380 enddef;
1381 vardef ddecimal primary p =
1382   decimal xpart p & " " & decimal ypart p
1383 enddef;
1384 vardef boundingbox primary p =
1385   if (path p) or (picture p) :
1386     llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1387   else :
1388     origin
1389   fi -- cycle
1390 enddef;
1391 fi
1392 def resolvedcolor(expr s) =
1393   runscript("return luamplib.shadecolor('"&s&"')")
1394 enddef;
1395 def colordecimals primary c =
1396   if cmykcolor c:
1397     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1398     decimal yellowpart c & ":" & decimal blackpart c
1399   elseif rgbcolor c:
1400     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1401   elseif string c:
1402     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1403   else:
1404     decimal c
1405   fi
1406 enddef;
1407 def externalfigure primary filename =
1408   draw rawtexttext("\includegraphics{"& filename &}")
1409 enddef;

```

```

1410 def TEX = texttext enddef;
1411 def mplibtexcolor primary c =
1412   runscript("return luamplib.gettexcolor('& c &'")
1413 enddef;
1414 def mplibrbgtexcolor primary c =
1415   runscript("return luamplib.gettexcolor('& c &', 'rgb'")
1416 enddef;
1417 def mplibgraphicstext primary t =
1418   begingroup;
1419   mplibgraphicstext_ (t)
1420 enddef;
1421 def mplibgraphicstext_ (expr t) text rest =
1422   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1423   fb, fc, dc, graphicstextpic, alsoordoublepath;
1424   picture graphicstextpic; graphicstextpic := nullpicture;
1425   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1426   let scale = scaled;
1427   def fakebold primary c = hide(fb:=c;) enddef;
1428   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1429   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1430   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1431   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1432   addto graphicstextpic alsoordoublepath (origin--cycle) rest; graphicstextpic:=nullpicture;
1433   def fakebold primary c = enddef;
1434   let fillcolor = fakebold; let drawcolor = fakebold;
1435   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1436   image(draw runscript("return luamplib.graphicstext([==["&t&"]==], "
1437     & decimal fb & ", "& fc & ", "& dc & "'") rest;))
1438 endgroup;
1439 enddef;
1440 def mplibglyph expr c of f =
1441   runscript (
1442     "return luamplib.glyph('"
1443     & if numeric f: decimal fi f
1444     & " ', '"
1445     & if numeric c: decimal fi c
1446     & " ')"
1447   )
1448 enddef;
1449 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1450 def mplibdrawglyph expr g =
1451   luamplib_tmp_num_ := 0;
1452   for item within g:
1453     fill pathpart item
1454     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1455   endfor
1456 enddef;
1457 let mplibfillglyph = mplibdrawglyph;
1458 def mplibstrokeyglyph expr g =

```

```

1459 luamplib_tmp_num_ := 0;
1460 for item within g:
1461   draw pathpart item
1462   if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1463 endfor
1464 enddef;
1465 def mplibfillandstrokeglyph expr g =
1466   luamplib_tmp_num_ := 0;
1467   for item within g:
1468     draw pathpart item withpostscript
1469     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1470   endfor
1471 enddef;
1472 def withmplibcolors (expr f, s) =
1473   runscript("return luamplib.fillandstrokecolor(' &
1474     if not string f: colordecimals fi f & '','' &
1475     if not string s: colordecimals fi s & '')")
1476 enddef;
1477 def withmplibopacities (expr a, f, s) =
1478   withprescript "tr_alternative=" & if numeric a: decimal fi a
1479   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1480 enddef;
1481 def mplib_do_outline_text_set_b (text f) (text d) text r =
1482   def mplib_do_outline_options_f = f enddef;
1483   def mplib_do_outline_options_d = d enddef;
1484   def mplib_do_outline_options_r = r enddef;
1485 enddef;
1486 def mplib_do_outline_text_set_f (text f) text r =
1487   def mplib_do_outline_options_f = f enddef;
1488   def mplib_do_outline_options_r = r enddef;
1489 enddef;
1490 def mplib_do_outline_text_set_u (text f) text r =
1491   def mplib_do_outline_options_f = f enddef;
1492 enddef;
1493 def mplib_do_outline_text_set_d (text d) text r =
1494   def mplib_do_outline_options_d = d enddef;
1495   def mplib_do_outline_options_r = r enddef;
1496 enddef;
1497 def mplib_do_outline_text_set_r (text d) (text f) text r =
1498   def mplib_do_outline_options_d = d enddef;
1499   def mplib_do_outline_options_f = f enddef;
1500   def mplib_do_outline_options_r = r enddef;
1501 enddef;
1502 def mplib_do_outline_text_set_n text r =
1503   def mplib_do_outline_options_r = r enddef;
1504 enddef;
1505 def mplib_do_outline_text_set_p = enddef;
1506 def mplib_fill_outline_text =
1507   for n=1 upto mpliboutlinenum:

```

```

1508     i:=0;
1509     for item within mpliboutlinepic[n]:
1510         i:=i+1;
1511         fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1512         if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1513     endfor
1514 endfor
1515 enddef;
1516 def mplib_draw_outline_text =
1517   for n=1 upto mpliboutlinenum:
1518     for item within mpliboutlinepic[n]:
1519       draw pathpart item mplib_do_outline_options_d;
1520     endfor
1521   endfor
1522 enddef;
1523 def mplib_filldraw_outline_text =
1524   for n=1 upto mpliboutlinenum:
1525     i:=0;
1526     for item within mpliboutlinepic[n]:
1527       i:=i+1;
1528       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1529         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1530       else:
1531         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1532       fi
1533     endfor
1534   endfor
1535 enddef;
1536 vardef mpliboutlinetext@# (expr t) text rest =
1537   save kind; string kind; kind := str @#;
1538   save i; numeric i;
1539   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1540   def mplib_do_outline_options_d = enddef;
1541   def mplib_do_outline_options_f = enddef;
1542   def mplib_do_outline_options_r = enddef;
1543   runscript("return luamplib.outlinetext[==["&t&"]==]");
1544   image ( addto currentpicture also image (
1545     if kind = "f":
1546       mplib_do_outline_text_set_f rest;
1547       mplib_fill_outline_text;
1548     elseif kind = "d":
1549       mplib_do_outline_text_set_d rest;
1550       mplib_draw_outline_text;
1551     elseif kind = "b":
1552       mplib_do_outline_text_set_b rest;
1553       mplib_fill_outline_text;
1554       mplib_draw_outline_text;
1555     elseif kind = "u":
1556       mplib_do_outline_text_set_u rest;

```

```

1557     mplib_filldraw_outline_text;
1558 elseif kind = "r":
1559     mplib_do_outline_text_set_r rest;
1560     mplib_draw_outline_text;
1561     mplib_fill_outline_text;
1562 elseif kind = "p":
1563     mplib_do_outline_text_set_p;
1564     mplib_draw_outline_text;
1565 else:
1566     mplib_do_outline_text_set_n rest;
1567     mplib_fill_outline_text;
1568 fi;
1569 ) mplib_do_outline_options_r; )
1570 enddef ;
1571 def withmppattern primary p =
1572   withprescript "mplibpattern=" & if numeric p: decimal fi p
1573 enddef;
1574 primarydef t withpattern p =
1575   image(
1576     if cycle t:
1577       fill
1578     else:
1579       draw
1580     fi
1581     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1582 enddef;
1583 vardef mplibtransformmatrix (text e) =
1584   save t; transform t;
1585   t = identity e;
1586   runscript("luamplib.transformmatrix = {"
1587     & decimal xpart t & ","
1588     & decimal ypart t & ","
1589     & decimal xpart t & ","
1590     & decimal ypart t & ","
1591     & decimal xpart t & ","
1592     & decimal ypart t & ","
1593     & "}");
1594 enddef;
1595 primarydef p withmaskinggroup s =
1596   if picture p:
1597     image(
1598       draw p;
1599       draw center p withprescript "mplibfadestate=stop";
1600     )
1601   else:
1602     p withprescript "mplibfadestate=stop"
1603   fi
1604   withprescript "mplibfadetype=masking"
1605   withprescript "mplibmaskname=" & s

```

```

1606 enddef;
1607 def withmaskingbgcolor expr c =
1608   withprescript "mplibmaskingbgcolor=" & colordecimals c
1609 enddef;
1610 primarydef p withfademethod s =
1611   if picture p:
1612     image(
1613       draw p;
1614       draw center p withprescript "mplibfadestate=stop";
1615     )
1616   else:
1617     p withprescript "mplibfadestate=stop"
1618   fi
1619   withprescript "mplibfadetype=" & s
1620   hide(mplib_shade_step := 1;)
1621   withprescript "sh_color_a=1"
1622   withprescript "sh_color_b=0"
1623   withprescript "mplibfadebbox=" &
1624     decimal (xpart llcorner p -1/4) & ":" &
1625     decimal (ypart llcorner p -1/4) & ":" &
1626     decimal (xpart urcorner p +1/4) & ":" &
1627     decimal (ypart urcorner p +1/4)
1628 enddef;
1629 def withfadevector (expr a,b) =
1630   withprescript "mplibfadevector=" &
1631     decimal xpart a & ":" &
1632     decimal ypart a & ":" &
1633     decimal xpart b & ":" &
1634     decimal ypart b
1635 enddef;
1636 let withfadecenter = withfadevector;
1637 def withfaderadius (expr a,b) =
1638   withprescript "mplibfaderadius=" &
1639     decimal a & ":" &
1640     decimal b
1641 enddef;
1642 def withfadebbox (expr a,b) =
1643   withprescript "mplibfadebbox=" &
1644     decimal xpart a & ":" &
1645     decimal ypart a & ":" &
1646     decimal xpart b & ":" &
1647     decimal ypart b
1648 enddef;
1649 primarydef p asgroup s =
1650   image(
1651     draw center p
1652     withprescript "mplibgroupbbox=" &
1653       decimal (xpart llcorner p -1/4) & ":" &
1654       decimal (ypart llcorner p -1/4) & ":" &

```

```

1655     decimal (xpart urcorner p +1/4) & ":" &
1656     decimal (ypart urcorner p +1/4)
1657     withprescript "gr_state=start"
1658     withprescript "gr_type=" & s;
1659     draw p withprescript "sh_in_xobj=yes";
1660     draw center p withprescript "gr_state=stop";
1661 )
1662 enddef;
1663 def withgroupbbox (expr a,b) =
1664   withprescript "mplibgroupbbox=" &
1665     decimal xpart a & ":" &
1666     decimal ypart a & ":" &
1667     decimal xpart b & ":" &
1668     decimal ypart b
1669 enddef;
1670 def withgroupname expr s =
1671   withprescript "mplibgroupname=" & s
1672 enddef;
1673 def usemplibgroup primary s =
1674   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s & "\endcsname}")
1675   shifted runscript("return luamplib.trgroupshifts['' & s & ''"]")
1676 enddef;
1677 path    mplib_shade_path ;
1678 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1679 numeric mplib_shade_fx, mplib_shade_fy ;
1680 numeric mplib_shade_lx, mplib_shade_ly ;
1681 numeric mplib_shade_nx, mplib_shade_ny ;
1682 numeric mplib_shade_dx, mplib_shade_dy ;
1683 numeric mplib_shade_tx, mplib_shade_ty ;
1684 primarydef p withshadingmethod m =
1685   p
1686   if picture p :
1687     withprescript "sh_operand_type=picture"
1688     if textual p or (length p > 1):
1689       withprescript "sh_transform=no"
1690       mplib_with_shade_method (boundingbox p, m)
1691     else:
1692       withprescript "sh_transform=yes"
1693       mplib_with_shade_method (pathpart p, m)
1694     fi
1695   else :
1696     withprescript "sh_transform=yes"
1697     mplib_with_shade_method (p, m)
1698   fi
1699 enddef;
1700 def mplib_with_shade_method (expr p, m) =
1701   hide(mplib_with_shade_method_analyze(p))
1702   withprescript "sh_domain=0 1"
1703   withprescript "sh_color=into"

```

```

1704 withprescript "sh_color_a=" & colordecimals white
1705 withprescript "sh_color_b=" & colordecimals black
1706 withprescript "sh_first=" & ddecimal point 0 of p
1707 withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1708 withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1709 if m = "linear" :
1710   withprescript "sh_type=linear"
1711   withprescript "sh_factor=1"
1712   withprescript "sh_center_a=" & ddecimal llcorner p
1713   withprescript "sh_center_b=" & ddecimal urcorner p
1714 elseif m = "coons":
1715   withprescript "sh_type=coons"
1716   withprescript "sh_transform=no"
1717 elseif m = "triangle":
1718   withprescript "sh_type=triangle"
1719   withprescript "sh_transform=no"
1720 elseif m = "lattice":
1721   withprescript "sh_type=lattice"
1722   withprescript "sh_transform=no"
1723 elseif m = "tensor":
1724   withprescript "sh_type=tensor"
1725   withprescript "sh_transform=no"
1726   withprescript "sh_tensor_path=" &
1727     ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1728     ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1729     ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1730     ddecimal 1/4[point 3 of p, point 1 of p]
1731 else :
1732   withprescript "sh_type=circular"
1733   withprescript "sh_factor=1.2"
1734   withprescript "sh_center_a=" & ddecimal center p
1735   withprescript "sh_center_b=" & ddecimal center p
1736   withprescript "sh_radius_a=" & decimal 0
1737   withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1738 fi
1739 enddef;
1740 def withlatticeverticesperrow expr n =
1741   withprescript "sh_lattice_perrow=" & decimal n
1742 enddef;
1743 def withlatticeverticesdata (text t) =
1744   hide(luamplib_tmp_num_ := 0;)
1745   withprescript "sh_lattice_data=" &
1746   for item = t:
1747     if odd(incr luamplib_tmp_num_):
1748       if pair item: ddecimal fi item & " " &
1749       fi
1750   endfor ""
1751   hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1752   for item = t:

```

```

1753     if not odd(incr luamplib_tmp_num_):
1754         withshadingstep( withshadingcolors(item,item) )
1755     fi
1756 endfor
1757 enddef;
1758 def withtrianglepatchinit (expr p, a, b, c) =
1759   if string p:
1760     withtrianglepatchnext(0, p , a)
1761     withtrianglepatchnext(0, "", b)
1762     withtrianglepatchnext(0, "", c)
1763   else:
1764     withtrianglepatchnext(0, point 0 of p, a)
1765     withtrianglepatchnext(0, point 1 of p, b)
1766     withtrianglepatchnext(0, point 2 of p, c)
1767   fi
1768 enddef;
1769 def withtrianglepatchnext (expr f, p, a) =
1770   withshadingstep(
1771     withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1772     withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1773       if string p: p else: ddecimal p fi
1774     withshadingcolors (a, a)
1775   )
1776 enddef;
1777 def withcoonspatchinit (expr p, a, b, c, d) =
1778   withshadingstep(
1779     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1780     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1781       if string p: p
1782     else:
1783       ddecimal point      0 of p & " " &
1784       ddecimal postcontrol 0 of p & " " &
1785       ddecimal precontrol  1 of p & " " &
1786       ddecimal point      1 of p & " " &
1787       ddecimal postcontrol 1 of p & " " &
1788       ddecimal precontrol  2 of p & " " &
1789       ddecimal point      2 of p & " " &
1790       ddecimal postcontrol 2 of p & " " &
1791       ddecimal precontrol  3 of p & " " &
1792       ddecimal point      3 of p & " " &
1793       ddecimal postcontrol 3 of p & " " &
1794       ddecimal precontrol  0 of p
1795     fi
1796     withshadingcolors (a, b)
1797   )
1798   withshadingstep ( withshadingcolors (c, d) )
1799 enddef;
1800 def withtensorpatchinit (expr p, q, a, b, c, d) =
1801   withshadingstep(

```

```

1802   withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1803   withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1804       if string p: p & " " & q
1805       else:
1806           ddecimal point      0 of p & " " &
1807           ddecimal postcontrol 0 of p & " " &
1808           ddecimal precontrol  1 of p & " " &
1809           ddecimal point      1 of p & " " &
1810           ddecimal postcontrol 1 of p & " " &
1811           ddecimal precontrol  2 of p & " " &
1812           ddecimal point      2 of p & " " &
1813           ddecimal postcontrol 2 of p & " " &
1814           ddecimal precontrol  3 of p & " " &
1815           ddecimal point      3 of p & " " &
1816           ddecimal postcontrol 3 of p & " " &
1817           ddecimal precontrol  0 of p & " " &
1818           ddecimal point      0 of q & " " &
1819           ddecimal point      1 of q & " " &
1820           ddecimal point      2 of q & " " &
1821           ddecimal point      3 of q
1822       fi
1823   withshadingcolors (a, b)
1824 )
1825 withshadingstep ( withshadingcolors (c, d) )
1826 enddef;
1827 def withcoonspatchnext (expr f, p, a, b) =
1828   withshadingstep(
1829     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1830     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1831       if string p: p
1832       else:
1833           ddecimal postcontrol 1 of p & " " &
1834           ddecimal precontrol  2 of p & " " &
1835           ddecimal point      2 of p & " " &
1836           ddecimal postcontrol 2 of p & " " &
1837           ddecimal precontrol  3 of p & " " &
1838           ddecimal point      3 of p & " " &
1839           ddecimal postcontrol 3 of p & " " &
1840           ddecimal precontrol  0 of p
1841       fi
1842     withshadingcolors (a, b)
1843   )
1844 enddef;
1845 def withtensorpatchnext (expr f, p, q, a, b) =
1846   withshadingstep(
1847     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1848     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1849       if string p: p & " " & q
1850       else:

```

```

1851      ddecimal postcontrol 1 of p & " " &
1852      ddecimal precontrol 2 of p & " " &
1853      ddecimal point 2 of p & " " &
1854      ddecimal postcontrol 2 of p & " " &
1855      ddecimal precontrol 3 of p & " " &
1856      ddecimal point 3 of p & " " &
1857      ddecimal postcontrol 3 of p & " " &
1858      ddecimal precontrol 0 of p & " " &
1859      ddecimal point 0 of q & " " &
1860      ddecimal point 1 of q & " " &
1861      ddecimal point 2 of q & " " &
1862      ddecimal point 3 of q
1863    fi
1864    withshadingcolors (a, b)
1865  )
1866 enddef;
1867 def mplib_with_shade_method_analyze(expr p) =
1868   mplib_shade_path := p ;
1869   mplib_shade_step := 1 ;
1870   mplib_shade_fx := xpart point 0 of p ;
1871   mplib_shade_fy := ypart point 0 of p ;
1872   mplib_shade_lx := mplib_shade_fx ;
1873   mplib_shade_ly := mplib_shade_fy ;
1874   mplib_shade_nx := 0 ;
1875   mplib_shade_ny := 0 ;
1876   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1877   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1878   for i=1 upto length(p) :
1879     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1880     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1881     if mplib_shade_tx > mplib_shade_dx :
1882       mplib_shade_nx := i + 1 ;
1883       mplib_shade_lx := xpart point i of p ;
1884       mplib_shade_dx := mplib_shade_tx ;
1885     fi ;
1886     if mplib_shade_ty > mplib_shade_dy :
1887       mplib_shade_ny := i + 1 ;
1888       mplib_shade_ly := ypart point i of p ;
1889       mplib_shade_dy := mplib_shade_ty ;
1890     fi ;
1891   endfor ;
1892 enddef;
1893 vardef mplib_max_radius(expr p) =
1894   max (
1895     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1896     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1897     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1898     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1899   )

```

```

1900 enddef;
1901 def withshadingstep (text t) =
1902   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1903   withprescript "sh_step=" & decimal mplib_shade_step
1904   t
1905 enddef;
1906 let withfadestep = withshadingstep;
1907 def withshadingradius expr a =
1908   withprescript "sh_radius_a=" & decimal (xpart a)
1909   withprescript "sh_radius_b=" & decimal (ypart a)
1910 enddef;
1911 def withshadingorigin expr a =
1912   withprescript "sh_center_a=" & ddecimal a
1913   withprescript "sh_center_b=" & ddecimal a
1914 enddef;
1915 def withshadingvector expr a =
1916   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1917   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1918 enddef;
1919 def withshadingdirection expr a =
1920   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1921   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1922 enddef;
1923 def withshadingtransform expr a =
1924   withprescript "sh_transform=" & a
1925 enddef;
1926 def withshadingcenter expr a =
1927   withprescript "sh_center_a=" & ddecimal (
1928     center mplib_shade_path shifted (
1929       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1930       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1931     )
1932   )
1933 enddef;
1934 def withshadingcenters (expr a, b) =
1935   withprescript "sh_center_a=" & ddecimal a
1936   withprescript "sh_center_b=" & ddecimal b
1937   withshadingtransform "no"
1938   withshadingfactor 1
1939 enddef;
1940 let withshadingpoints = withshadingcenters;
1941 def withshadingextend (expr a, b) =
1942   withprescript "sh_extend=" &
1943     if a: "true" else: "false" fi & " " &
1944     if b: "true" else: "false" fi
1945 enddef;
1946 let withfadeextend = withshadingextend;
1947 def withshadingdomain expr d =
1948   withprescript "sh_domain=" & ddecimal d

```

```

1949 enddef;
1950 def withshadingfactor expr f =
1951   withprescript "sh_factor=" & decimal f
1952 enddef;
1953 def withshadingfraction expr a =
1954   if mplib_shade_step > 0 :
1955     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1956   fi
1957 enddef;
1958 let withfadefraction = withshadingfraction;
1959 def withshadingcolors (expr a, b) =
1960   if mplib_shade_step > 0 :
1961     withprescript "sh_color=into"
1962     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1963     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1964   else :
1965     withprescript "sh_color=into"
1966     withprescript "sh_color_a=" & colordecimals a
1967     withprescript "sh_color_b=" & colordecimals b
1968   fi
1969 enddef;
1970 let withfadeopacity = withshadingcolors;
1971 def withshadingstroke expr a =
1972   withprescript "sh_stroking=" & a
1973 enddef;
1974 def withshadingmatrix expr s =
1975   withprescript "sh_matrix=" & s
1976 enddef;
1977 let withfadematrix = withshadingmatrix;
1978 def mpliblength primary t =
1979   runscript("return utf8.len[==[" & t & "]==]")
1980 enddef;
1981 def mplibsubstring expr p of t =
1982   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1983     & decimal xpart p & ",",
1984     & decimal ypart p & ")")
1985 enddef;
1986 def mlibuclength primary t =
1987   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
1988 enddef;
1989 def mlibucsubstring expr p of t =
1990   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1991     & decimal xpart p & ",",
1992     & decimal ypart p & ",true)")
1993 enddef;
1994 ]],
1995 legacyverbatimtex = [[
1996 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1997 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;

```

```

1998 let VerbatimTeX = specialVerbatimTeX;
1999 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
2000 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
2001 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
2002 "runscript(" &ditto&
2003 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
2004 "luamplib.in_the_fig=false" &ditto& ");";
2005 ]],
2006 texttextlabel = [[
2007 let luampliboriginalinfont = infont;
2008 primarydef s infont f =
2009   if (s < char 32)
2010     or (s = char 35) % #
2011     or (s = char 36) % $
2012     or (s = char 37) % %
2013     or (s = char 38) % &
2014     or (s = char 92) % \
2015     or (s = char 94) % ^
2016     or (s = char 95) % _
2017     or (s = char 123) % {
2018     or (s = char 125) % }
2019     or (s = char 126) % ~
2020     or (s = char 127) :
2021     s luampliboriginalinfont f
2022   else :
2023     rawtexttext(s)
2024   fi
2025 enddef;
2026 def fontsize expr f =
2027   begingroup
2028   save size; numeric size;
2029   size := mplibdimen("1em");
2030   if size = 0: 10pt else: size fi
2031 endgroup
2032 enddef;
2033 ]],
2034 }
2035

```

process_mplibcode

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

```

2036 luamplib.verbatiminput = false
2037 luamplib.everymplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2038 luamplib.everyendmplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2039 function luamplib.process_mplibcode (data, instancename)
2040   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2041 if luamplib.legacyverbatim then
2042   luamplib.figid, tex_code_pre_mplib = 1, {}

```

```

2043 end
2044 local everymplib = luamplib.everymplib[instancename]
2045 local everyendmplib = luamplib.everyendmplib[instancename]
2046 data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
2047 :gsub("\r","\n")

```

These five lines are needed for mplibverbatim mode.

```

2048 if luamplib.verbatiminput then
2049   data = data:gsub("\mpcolor%{s+}(.-%b{ })","mplibcolor(\"%1\")")
2050   :gsub("\mpdim%{s+}(%b{ })", "mplibdimen(\"%1\")")
2051   :gsub("\mpdim%{s+}(\%a+)", "mplibdimen(\"%1\")")
2052   :gsub(btex_etex, "btex %1 etex ")
2053   :gsub(verbatimtex_etex, "verbatimtex %1 etex;")
2054 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.

```

2055   local t = { } -- to store btex, verbatimtex, string
2056   data = data:gsub(btex_etex, function(str)
2057     t[#t+1] = str
2058     return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
2059   end)
2060   :gsub(verbatimtex_etex, function(str)
2061     t[#t+1] = str
2062     return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
2063   end)
2064   :gsub('\"(.-)\"', function(str)
2065     t[#t+1] = str
2066     return format('\"\\unexpanded{!l!u!a!%s!m!p!l!}\"', #t)
2067   end)
2068   :gsub("\\\\%", "\\0PerCent\0")
2069   :gsub("%%.-\\n", "\\n")
2070   :gsub("%zPerCent%z", "\\%")
2071   run_tex_code(format("\mplibtmptoks\\expandafter{\\expanded{%s}}",data))
2072   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

2073   :gsub("##", "#")
2074   :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2075 end
2076 process(data, instancename)
2077 end
2078

```

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.

```

2079 local figcontents = { post = { } }
2080 local function put2output(a,...)
2081   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a

```

```

2082 end
2083 local function pdf_startfigure(n,llx,lly,urx,ury)
2084   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2085 end
2086 local function pdf_stopfigure()
2087   put2output("\mplibstoptoPDF")
2088 end

```

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

```

2089 local function pdf_literalcode (...)
2090   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2091 end
2092 local start_pdf_code = pdfmode
2093   and function() pdf_literalcode"q" end
2094   or function() put2output"\special{pdf:bcontent}" end
2095 local stop_pdf_code = pdfmode
2096   and function() pdf_literalcode"Q" end
2097   or function() put2output"\special{pdf:econtent}" end
2098

```

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

```

2099 local function put_tex_boxes (object,prescript)
2100   local box = prescript.mplibtexboxid:explode:""
2101   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2102   if n and tw and th then
2103     local op = object.path
2104     local first, second, fourth = op[1], op[2], op[4]
2105     local tx, ty = first.x_coord, first.y_coord
2106     local sx, rx, ry, sy = 1, 0, 0, 1
2107     if tw ~= 0 then
2108       sx = (second.x_coord - tx)/tw
2109       rx = (second.y_coord - ty)/tw
2110       if sx == 0 then sx = 0.00001 end
2111     end
2112     if th ~= 0 then
2113       sy = (fourth.y_coord - ty)/th
2114       ry = (fourth.x_coord - tx)/th
2115       if sy == 0 then sy = 0.00001 end
2116     end
2117

```

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

```

2118   local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2119   put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2120 end
2121 end
2122

```

Colors

```

2123 local function do_preobj_CR(object,prescript)
2124   if object.postscript == "collect" then return end

```

```

2125 local override = prescript and prescript.mpliboverridecolor
2126 if override then
2127     pdf_literalcode(override)
2128 else
2129     local cs = object.color
2130     if cs and #cs > 0 then
2131         pdf_literalcode(luamplib.colorconverter(cs))
2132     end
2133 end
2134 end
2135

```

For transparency, shading, fading, and pattern

```

2136 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2137 local pdfobjs, pdfetcs = {}, {}
2138 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2139 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2140 pdfetcs.pgfcspaces = "pgf@sys@addpdfresource@colorspaces@plain"
2141 local update_pdfobjs
2142 if pdfmode then
2143     function update_pdfobjs (os, stream)
2144         local key = os
2145         if stream then key = key..stream end
2146         local on = key and pdfobjs[key]
2147         if on then
2148             return on,false
2149         end
2150         if stream then
2151             on = pdf.immediateobj("stream",stream,os)
2152         elseif os then
2153             on = pdf.immediateobj(os)
2154         else
2155             on = pdf.reserveobj()
2156         end
2157         if key then
2158             pdfobjs[key] = on
2159         end
2160         return on,true
2161     end
2162 else
2163     function update_pdfobjs (os, stream, hex)
2164         local key = os
2165         if stream then key = key..stream end
2166         local on = key and pdfobjs[key]
2167         if on then
2168             return on,false
2169         end
2170         on = pdfetcs.cnt or 1
2171         if stream then

```

```

2172     if hex then
2173         texsprint(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2174     else
2175         texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2176     end
2177 elseif os then
2178     texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2179 else
2180     texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2181 end
2182 pdfetcs.cnt = on + 1
2183 if key then
2184     pdfobjs[key] = on
2185 end
2186 return on,true
2187 end
2188 end
2189 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2190 if pdfmode then
2191     pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2192     local getpagers = pdfetcs.getpagers
2193     local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2194     local initialize_resources = function(name)
2195         local tabname = format("%s_res",name)
2196         pdfetcs[tabname] = { }
2197         if luatexbase.callbacktypes.finish_pdffile then -- ltuatex
2198             local obj = pdf.reserveobj()
2199             setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2200             luatexbase.add_to_callback("finish_pdffile", function()
2201                 pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
2202             end,
2203             format("luamplib.%s.finish_pdffile",name))
2204         end
2205     end
2206     pdfetcs.fallback_update_resources = function(name, res)
2207         local tabname = format("%s_res",name)
2208         if not pdfetcs[tabname] then
2209             initialize_resources(name)
2210         end
2211         if luatexbase.callbacktypes.finish_pdffile then
2212             local t = pdfetcs[tabname]
2213             t[#t+1] = res
2214         else
2215             local tpr, n = getpagers() or "", 0
2216             tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2217             if n == 0 then
2218                 tpr = format("%s/%s<<%s>>", tpr, name, res)
2219             end
2220             setpagers(tpr)

```

```

2221     end
2222 end
2223 else
2224   texsprint {
2225     "\\luamplibatfirstshipout{",
2226     "\\special{pdf:obj @MPLibTr<<>>}",
2227     "\\special{pdf:obj @MPLibSh<<>>}",
2228     "\\special{pdf:obj @MPLibCS<<>>}",
2229     "\\special{pdf:obj @MPLibPt<<>>}}",
2230   }
2231   pdfetcs.fallback_update_resources = function (name,res,obj)
2232     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2233     local tabname = format("%s_res",name)
2234     if not pdfetcs[tabname] then
2235       texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2236       pdfetcs[tabname] = { }
2237     end
2238     tableinsert(pdfetcs[tabname], res)
2239   end
2240 end
2241

```

Transparency

```

2242 local function add_extgs_resources (on, new)
2243   local key = format("MPLibTr%s", on)
2244   if new then
2245     local val = format(pdfetcs.resfmt, on)
2246     if pdfmanagement then
2247       texsprint {
2248         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2249       }
2250     else
2251       local tr = format("/%s %s", key, val)
2252       if is_defined(pdfetcs.pgfextgs) then
2253         texsprint { "\\csname ", pdfetcs.pgfextgs, "\\endcsname{" , tr, "}" }
2254       elseif is_defined"TRP@list" then
2255         texsprint(catat11,{
2256           [[\if@files\immediate\write\auxout{]],
2257           [[\string\g@addto@macro\string\TRP@list{]],
2258           tr,
2259           [[}]\fi]],
2260         })
2261         if not get_macro"TRP@list":find(tr) then
2262           texsprint(catat11,[[\global\TRP@reruntrue]])
2263         end
2264       else
2265         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2266       end
2267     end

```

```

2268 end
2269 return key
2270 end
2271
2272 local do_preobj_TR
2273 do
2274   local transparency_modes = {
2275     [0] = "Normal",
2276     "Normal",      "Multiply",    "Screen",      "Overlay",
2277     "SoftLight",   "HardLight",   "ColorDodge",  "ColorBurn",
2278     "Darken",      "Lighten",    "Difference",  "Exclusion",
2279     "Hue",          "Saturation", "Color",       "Luminosity",
2280     "Compatible",
2281     normal      = "Normal",    multiply = "Multiply",    screen = "Screen",
2282     overlay     = "Overlay",   softlight = "SoftLight",  hardlight = "HardLight",
2283     colordodge  = "ColorDodge", colorburn = "ColorBurn",  darken = "Darken",
2284     lighten     = "Lighten",   difference = "Difference", exclusion = "Exclusion",
2285     hue         = "Hue",       saturation = "Saturation", color = "Color",
2286     luminosity  = "Luminosity", compatible = "Compatible",
2287   }
2288   function do_preobj_TR(object,prescript)
2289     if object.postscript == "collect" then return end
2290     local opaq = prescript and prescript.tr_transparency
2291     if not opaq then return end
2292
2293     local key, on, os, new
2294     local mode = prescript.tr_alternative or 1
2295     mode = transparency_modes[tonumber(mode) or mode:lower()]
2296     if not mode then
2297       mode = prescript.tr_alternative
2298       warn("unsupported blend mode: '%s'", mode)
2299     end
2300     opaq = opaq:explode":"
2301     for i,v in ipairs(opaq) do
2302       opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2303     end
2304     for i,v in ipairs{ {mode,opaq[1],opaq[2] or opaq[1]},{"Normal",1,1} } do
2305       os = format("</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2306       on, new = update_pdfobjs(os)
2307       key = add_extgs_resources(on,new)
2308       if i == 1 then
2309         pdf_literalcode("/%s gs",key)
2310       else
2311         return format("/%s gs",key)
2312       end
2313     end
2314   end
2315 end
2316

```

Shading with *metafun* format.

```

2317 local function add_shading_resources (on, new)
2318   if new then
2319     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2320     if pdfmanagement then
2321       texsprint {
2322         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2323       }
2324     else
2325       local res = format("/%s %s", key, val)
2326       pdfetcs.fallback_update_resources("Shading", res, "@MPlibSh")
2327     end
2328   end
2329 end
2330 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2331   for _, v in ipairs{ca, cb} do
2332     for i, vv in ipairs(v) do
2333       for ii, vvv in ipairs(vv) do
2334         v[i][ii] = tonumber(vvv) and format("%.3f", vvv) or vvv
2335       end
2336     end
2337   end
2338   local fun2fmt, os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2339   if steps > 1 then
2340     local list, bounds, encode = { }, { }, { }
2341     for i=1, steps do
2342       if i < steps then
2343         bounds[i] = format("%.3f", fractions[i] or 1)
2344       end
2345       encode[2*i-1] = 0
2346       encode[2*i] = 1
2347       os = fun2fmt:format(domain, tableconcat(ca[i], ' '), tableconcat(cb[i], ' '))
2348       :gsub(decimals, rmzeros)
2349       list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2350     end
2351     os = tableconcat {
2352       "<</FunctionType 3",
2353       format("/Bounds[%s]", tableconcat(bounds, ' ')),
2354       format("/Encode[%s]", tableconcat(encode, ' ')),
2355       format("/Functions[%s]", tableconcat(list, ' ')),
2356       format("/Domain[%s]>>", domain),
2357     } :gsub(decimals, rmzeros)
2358   else
2359     os = fun2fmt:format(domain, tableconcat(ca[1], ' '), tableconcat(cb[1], ' '))
2360     :gsub(decimals, rmzeros)
2361   end
2362   local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2363   os = tableconcat {
2364     format("<</ShadingType %i", shtype),

```

```

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

```

```

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

```

```

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

```

```

2512     for i = 1, steps do
2513         local t = prescript["sh_triangle_vertex_" .. i]:explode()
2514         if #t == 2 then
2515             coords[#coords+1] = { t[1], t[2] }
2516         elseif #t == 6 then
2517             coords[#coords+1] = { t[1], t[2] }
2518             coords[#coords+1] = { t[3], t[4] }
2519             coords[#coords+1] = { t[5], t[6] }
2520         end
2521         if not ca[i] then err"colorspace mismatch?" end
2522         colors[#colors+1] = { ca[i] }
2523         edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])
2524     end
2525 end
2526
2527 local stream, xmin, ymin, wd, ht, devicen = { }
2528 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2529 colors, devicen = colors_ff(colors)
2530 for i = 1, #coords do
2531     stream[#stream+1] = string.char(edges[i])
2532     stream[#stream+1] = tableconcat(coords[i])
2533     stream[#stream+1] = tableconcat(colors[i])
2534 end
2535
2536 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2537 local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2538 return on, matrix
2539 end
2540 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2541     local tensor = prescript.sh_type == "tensor"
2542     local steps = tonumber(prescript.sh_step) or 0
2543     local coords, colors, edges = { }, { }, { }
2544     if steps < 2 then
2545         local path, t = object.path, { }
2546         for i = 1, 4 do
2547             t[#t+1] = path[i].x_coord
2548             t[#t+1] = path[i].y_coord
2549             t[#t+1] = path[i].right_x
2550             t[#t+1] = path[i].right_y
2551             local j = i == 4 and 1 or i+1
2552             t[#t+1] = path[j].left_x
2553             t[#t+1] = path[j].left_y
2554         end
2555         if tensor then
2556             t = table.move(prescript.sh_tensor_path:explode(), 1, 8, #t+1, t)
2557         end
2558         coords = { t }
2559         colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2560         edges = { 0 }

```

```

2561     colorspace = "/DeviceRGB"
2562 else
2563     for i = 1, steps do
2564         local edge = tonumber(prescript["sh_coons_edge_"..i])
2565         if edge then
2566             edges[#edges+1] = edge
2567             coords[#coords+1] = prescript["sh_coons_path_"..i]:explode()
2568             if edge == 0 then
2569                 if not (ca[i] and cb[i] and ca[i+1] and cb[i+1]) then err"colorspace mismatch?" end
2570                 colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2571             else
2572                 if not (ca[i] and cb[i]) then err"colorspace mismatch?" end
2573                 colors[#colors+1] = { ca[i], cb[i] }
2574             end
2575         end
2576     end
2577 end
2578
2579 local stream, xmin, ymin, wd, ht, devicen = { }
2580 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2581 colors, devicen = colors_ff(colors)
2582 for i = 1, #coords do
2583     stream[#stream+1] = string.char(edges[i])
2584     stream[#stream+1] = tableconcat(coords[i])
2585     stream[#stream+1] = tableconcat(colors[i])
2586 end
2587
2588 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin):gsub(decimals,rmzeros)
2589 local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2590 return on, matrix
2591 end
2592 function do_preobj_SH(object, prescript)
2593     local shade_no
2594     local sh_type = prescript and prescript.sh_type
2595     if not sh_type then return end
2596
2597     local domain = prescript.sh_domain or "0 1"
2598     local centera = (prescript.sh_center_a or "0 0"):explode()
2599     local centerb = (prescript.sh_center_b or "0 0"):explode()
2600     local transform = prescript.sh_transform == "yes"
2601     local sx,sy,sr,dx,dy = 1,1,1,0,0
2602     if transform then
2603         local first = (prescript.sh_first or "0 0"):explode()
2604         local setx = (prescript.sh_set_x or "0 0"):explode()
2605         local sety = (prescript.sh_set_y or "0 0"):explode()
2606         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2607         if x ~= 0 and y ~= 0 then
2608             local path = object.path
2609             local path1x = path[1].x_coord

```

```

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

```

```

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

```

```

2708     shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2709 elseif sh_type == "triangle" then
2710     shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2711 elseif sh_type == "lattice" then
2712     shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2713 else
2714     err"unknown shading type"
2715 end
2716
2717 local matrix = prescript.sh_matrix
2718 if matrix and matrix:find"%a" then
2719     local t = get_mp_matrix(matrix)
2720     matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2721 end
2722 if mesh_matrix and matrix then
2723     local a, b = mesh_matrix:explode(), matrix:explode()
2724     matrix = format("%f %f %f %f %f %f",
2725         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],
2726         a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)
2727 end
2728
2729 return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix
2730 end
2731 end
2732

```

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

```

2733 local function add_pattern_resources (key, val)
2734 if pdfmanagement then
2735     texsprint {
2736         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2737     }
2738 else
2739     local res = format("/%s %s", key, val)
2740     if is_defined(pdfetcs.pgfpattern) then
2741         texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2742     else
2743         pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2744     end
2745 end
2746 end
2747 if pdfmode then
2748     function luamplib.dolatelua (on, os, matrix)
2749         local h, v = pdf.getpos()
2750         local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2751         matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2752         :gsub(decimals,rmzeros)
2753         pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2754         pdf.refobj(on)

```

```

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

```

```

2804     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2805     local f = ioopen(name)
2806     if f then
2807         for line in f:lines() do
2808             local t = line:explode()
2809             pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2810         end
2811         f:close()
2812     end
2813 end

```

This seems to be needed for proper functioning:

```

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

2814     local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2815     local mt = matrix:explode()
2816     matrix = format("%s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2817     texsprint{ "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2818                 format(" <<%s/Matrix[%s]>>}", os, matrix) }
2819     put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2820                xobj and ("%s',[[%s]]"):format(xobj[1], xobj[2]))
2821 end
2822 ::skip_latelua::
2823 local key, val = format("MPlibPt%s", on), format(pdfetcs.resfmt, on)
2824 add_pattern_resources(key,val)
2825 pdf_literalcode("/Pattern cs/%s scn", key)

```

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

```

2826 prescript.sh_type = nil
2827 end
2828

```

Tiling Patterns

```

2829 pdfetcs.patterns = { }
2830 local function gather_resources (optres, ispattern)
2831     local t = { }
2832     if pdfmanagement then
2833         for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2834             local mytoks
2835             run_tex_code ({
2836                 "\\mplibmtoks\\expanded{{" ,
2837                 "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/" , v , "}" ,
2838                 "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/" , v , "}}", "}" ,
2839             }, ccexplat)
2840             mytoks = texgettoks"mplibmtoks"
2841             if not pdfmode then
2842                 mytoks = mytoks:gsub("\\str_convert_pdfname:n%s*{(.-)}", "%1") -- why not expanded?
2843             end

```

```

2844     mytoks = mytoks and mytoks:gsub("^%s*(.)%s*$", "%1")
2845     if mytoks and mytoks ~= "" then
2846         t[#t+1] = ("%s<<%s>>"):format(v, mytoks)
2847     end
2848 end
2849 elseif is_defined(pdfetcs.pgftextgs) then
2850     run_tex_code"\relax" -- flush tex.sprint queue
2851     if pdfmode then
2852         for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2853                             ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2854                             Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2855             local res = (get_macro(v) or ""):gsub("^%s*(.)%s*$", "%1")
2856             if res ~= "" then
2857                 t[#t+1] = ("%s<<%s>>"):format(k, res )
2858             end
2859         end
2860     else
2861         local abc = get_macro"pgfutil@abc" or ""
2862         for k,v in pairs { ExtGState = "@pgftextgs",
2863                             ColorSpace = "@pgfcolorspaces",
2864                             Pattern = "@pgfpatterns", } do
2865             local tt = { }
2866             for vv in abc:gmatch( v .. "%s*(%b<>)" ) do
2867                 tt[#tt+1] = vv:match("^<<%s*(.)%s*>>$")
2868             end
2869             if #tt > 0 then
2870                 t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2871             end
2872         end
2873     end

```

We still have to deal with Shading resources.

```

2874     if luatexbase.callbacktypes.finish_pdffile then
2875         if pdfetcs.Shading_res then
2876             t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2877         end
2878     else
2879         local res = pdfetcs.getpageres()
2880         res = res and res:match"/Shading%s*%b<>"
2881         if res then
2882             t[#t+1] = res
2883         end
2884     end
2885 else
2886     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

```

2887     warn"transparent package is not fully functional without pdfmanagement code."
2888 end

```

```

2889   if luatexbase.callbacktypes.finish_pdffile then
2890     for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2891       local tt = pdfetcs[v.."_res"]
2892       if tt then
2893         t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2894       end
2895     end
2896   else
2897     local res = pdfetcs.getpageres()
2898     if res then
2899       t[#t+1] = res
2900     end
2901   end
2902 end
2903 local result = tableconcat(t)
2904 if optres ~= "" then
2905   for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2906     local res = optres:match("/"..v.."%"s*%b<>")
2907     if res then
2908       if result:find("/"..v) then
2909         res = res:match("<<(.+)>>$")
2910         result = result:gsub("/"..v.."%"s*<<,""%1"..res, 1)
2911       else
2912         result = result .. res
2913       end
2914     end
2915   end
2916 end
2917 return result
2918 end
2919 function luamplib.registerpattern ( boxid, name, opts )
2920   local box = texgetbox(boxid)
2921   local wd = format("%.3f",box.width/factor)
2922   local hd = format("%.3f",(box.height+box.depth)/factor)
2923   info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2924   if opts.xstep == 0 then opts.xstep = nil end
2925   if opts.ystep == 0 then opts.ystep = nil end
2926   if opts.colored == nil then
2927     opts.colored = opts.coloured
2928     if opts.colored == nil then
2929       opts.colored = true
2930     end
2931   end
2932   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2933   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2934   if opts.matrix and opts.matrix:find"%a" then
2935     local t = get_mp_matrix(opts.matrix)
2936     opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2937     opts.xshift = opts.xshift or format("%f",t[5])

```

```

2938     opts.yshift = opts.yshift or format("%f",t[6])
2939 end
2940 local attr = {
2941     "/Type/Pattern",
2942     "/PatternType 1",
2943     format("/PaintType %i", opts.colored and 1 or 2),
2944     "/TilingType 2",
2945     format("/XStep %s", opts.xstep or wd),
2946     format("/YStep %s", opts.ystep or hd),
2947     format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2948 }
2949 local optres = opts.resources or ""
2950 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2951 local patterns = pdfetcs.patterns
2952 if pdfmode then
2953     if opts.bbox then
2954         attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2955     end
2956     attr = tableconcat(attr) :gsub(decimals,rmzeros)
2957     local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2958     patterns[name] = { id = index, colored = opts.colored }
2959 else
2960     local cnt = #patterns + 1
2961     local objname = "@mplibpattern" .. cnt
2962     local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2963     texsprint {
2964         "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2965         "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2966         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2967         "\\special{pdf:bcontent}}",
2968         "\\special{pdf:bxobj ", objname, " ", metric, "}",
2969         "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2970         "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2971         "\\special{pdf:put @resources <<", optres, ">>}",
2972         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2973         "\\special{pdf:econtent}}",
2974     }
2975     patterns[cnt] = objname
2976     patterns[name] = { id = cnt, colored = opts.colored }
2977     patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
2978 end
2979 end
2980
2981 local do_preobj_PAT
2982 do
2983     local function pattern_colorspace (cs)
2984         local on, new = update_pdfobjs(format("/Pattern %s]", cs))
2985         if new then
2986             local key, val = format("MPLibCS%i",on), format(pdfetcs.resfmt,on)

```

```

2987     if pdfmanagement then
2988         texsprint {
2989             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
2990         }
2991     else
2992         local res = format("/%s %s", key, val)
2993         if is_defined(pdfetcs.pgfcolorspace) then
2994             texsprint { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
2995         else
2996             pdfetcs.fallback_update_resources("ColorSpace",res,"@MPlibCS")
2997         end
2998     end
2999 end
3000 return on
3001 end
3002 function do_preobj_PAT(object, prescript)
3003     local name = prescript and prescript.mplibpattern
3004     if not name then return end
3005     local patterns = pdfetcs.patterns
3006     local patt = patterns[name]
3007     local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3008     local key = format("MPlibPt%s",index)
3009     if patt.colored then
3010         pdf_literalcode("/Pattern cs /%s scn", key)
3011     else
3012         local color = prescript.mpliboverridecolor
3013         if not color then
3014             local t = object.color
3015             color = t and #t>0 and luamplib.colorconverter(t)
3016         end
3017         if not color then return end
3018         local cs
3019         if color:find" cs " then
3020             local name
3021             name, color = color:match"^/(.-) cs (.-) sc"
3022             if pdfmode then
3023                 cs = format("%s 0 R", ltx.pdf.object_id( name ))
3024                 if cs == "0 0 R" then -- assumes colorspace.sty
3025                     _, cs = get_spc_name_obj("/"..name)
3026                 end
3027             else
3028                 run_tex_code({ "\\mplibtmp toks\\expanded{\\pdf_object_ref:n{", name, "}}}" }, ccexplat)
3029                 cs = texgettoks"mplibtmp toks"
3030             end
3031         else
3032             local t = colorsplit(color)
3033             cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
3034             color = tableconcat(t, " ")
3035         end

```

```

3036     pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3037 end
3038 if not patt.done then
3039     local val = pdfmode and format("%s 0 R",index) or patterns[index]
3040     add_pattern_resources(key,val)
3041 end
3042 patt.done = true
3043 end
3044 end
3045

```

Fading

```

3046 pdfetcs.fading = { }
3047 local function do_preobj_FADE (object, prescript)
3048     local fd_type = prescript and prescript.mplibfadetype
3049     local fd_stop = prescript and prescript.mplibfadestate
3050     if not fd_type then
3051         return fd_stop -- returns "stop" (if picture) or nil
3052     end
3053     local on, os, new
3054     if fd_type == "masking" then
3055         local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3056         on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (-)}")
3057         local bc = prescript.mplibmaskingbgcolor
3058         bc = bc and bc:gsub(":", " ")
3059         bc = bc and ("BC[%s]"):format(bc):gsub(decimals,rmzeros) or ""
3060         os = format("<</SMask<</S/Luminosity/G %s>>>>",
3061             pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3062     else
3063         local bbox = prescript.mplibfadebbox:explode":"
3064         local dx, dy = -bbox[1], -bbox[2]
3065         local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
3066         if not vec then
3067             if fd_type == "linear" then
3068                 vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3069             else
3070                 local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3071                 vec = {centerx, centery, centerx, centery} -- center for both circles
3072             end
3073         end
3074         local coords = { vec[1], vec[2], vec[3], vec[4] }
3075         if fd_type == "linear" then
3076             coords = format("%f %f %f %f", tableunpack(coords))
3077         elseif fd_type == "circular" then
3078             local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3079             local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":"
3080             tableinsert(coords, 3, radius[1])
3081             tableinsert(coords, radius[2])
3082             coords = format("%f %f %f %f %f %f", tableunpack(coords))

```

```

3083     else
3084         err("unknown fading method '%s'", fd_type)
3085     end
3086     fd_type = fd_type == "linear" and 2 or 3
3087     local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3088     local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":" }
3089     local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":" }
3090     if steps > 1 then
3091         fractions = { prescript.sh_fraction_1 or 0 }
3092         for i=2,steps do
3093             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3094             ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":"
3095             cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":"
3096         end
3097     end
3098     local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3099     matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3100     matrix[5] = matrix[5] + dx
3101     matrix[6] = matrix[6] + dy
3102     matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3103     on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3104     os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3105     on = update_pdfobjs(os)
3106     bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3107     local streamtext = format("q /Pattern cs/MPlibFd%s scn %s re f Q", on, bbox)
3108     :gsub(decimals,rmzeros)
3109     os = format("<</Pattern<</MPlibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3110     on = update_pdfobjs(os)
3111     local resources = format(pdfetcs.resfmt, on)
3112     on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
3113     local attr = tableconcat{
3114         "/Subtype/Form",
3115         "/BBox[" .. bbox .. "]",
3116         "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
3117         "/Resources " .. resources,
3118         "/Group " .. format(pdfetcs.resfmt, on),
3119     } :gsub(decimals,rmzeros)
3120     on = update_pdfobjs(attr, streamtext)
3121     os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
3122     end
3123     on, new = update_pdfobjs(os)
3124     local key = add_extgs_resources(on,new)
3125     start_pdf_code()
3126     pdf_literalcode("/%s gs", key)
3127     if fd_stop then return "standalone" end
3128     return "start"
3129 end
3130

```

Transparency Group

```

3131 pdfetcs.tr_group = { shifts = { } }
3132 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3133 local function do_preobj_GRP (object, prescript)
3134   local grstate = prescript and prescript.gr_state
3135   if not grstate then return end
3136   local trgroup = pdfetcs.tr_group
3137   if grstate == "start" then
3138     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3139     trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3140     trgroup.wrapped = false
3141     for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+)") do
3142       trgroup[v] = true
3143     end
3144     trgroup.bbox = prescript.mplibgroupbbox:explode":"
3145     put2output[[\begingroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3146   elseif grstate == "stop" then
3147     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3148     put2output(tableconcat{
3149       "\\egroup",
3150       format("\\wd\mplibscratchbox %fbp", urx-llx),
3151       format("\\ht\mplibscratchbox %fbp", ury-lly),
3152       "\\dp\mplibscratchbox 0pt",
3153     })
3154     local grattr
3155     if trgroup.off then
3156       grattr = ""
3157     else
3158       local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>",
3159                                         trgroup.isolated, trgroup.knockout))
3160       grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3161     end
3162     local res = gather_resources("")
3163     local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3164     if pdfmode then
3165       if trgroup.wrapped then
3166         put2output(tableconcat{
3167           "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3168           "/BBox[" .. bbox .. "] } resources{", res, "}" .. "\\mplibscratchbox",
3169           "\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}",
3170         })
3171       end
3172       put2output(tableconcat{
3173         "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3174         "/BBox[" .. bbox .. "], grattr, " } resources{", res, "}" .. "\\mplibscratchbox",
3175         "\\luamplibtagasgroupput{" .. trgroup.name .. "}",
3176         [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
3177         [[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]],
3178         [[\box\mplibscratchbox]],

```

```

3179     "\\endgroup",
3180     "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{",
3181     "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3182     "\\useboxresource \\the\\lastsavedboxresourceindex",
3183     "\\}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3184     "\\box\\mplibscratchbox}",
3185   })
3186   else
3187     if trgroup.wrapped then
3188       trgroup.cnt = (trgroup.cnt or 0) + 1
3189       local objname = format("@mplibtrgr%s", trgroup.cnt)
3190       put2output(tableconcat{
3191         "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3192         "\\unhbox\\mplibscratchbox",
3193         "\\special{pdf:put @resources <<", res, ">>}",
3194         "\\special{pdf:exobj}",
3195         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}",
3196       })
3197     end
3198     trgroup.cnt = (trgroup.cnt or 0) + 1
3199     local objname = format("@mplibtrgr%s", trgroup.cnt)
3200     put2output(tableconcat{
3201       "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3202       "\\unhbox\\mplibscratchbox",
3203       "\\special{pdf:put @resources <<", res, ">>}",
3204       "\\special{pdf:exobj <<", grattr, ">>}",
3205       "\\luamplibtagasgroupput{",trgroup.name,"}",
3206       "\\special{pdf:uxobj ", objname, "}",
3207       "\\}\\endgroup",
3208     })
3209     token.set_macro("luamplib.group.".trgroup.name, tableconcat{
3210       "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3211       "\\special{pdf:uxobj ", objname, "}",
3212       "\\}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3213       "\\box\\mplibscratchbox",
3214     }, "global")
3215   end
3216   trgroup.shifts[trgroup.name] = { llx, lly }
3217 end
3218 return grstate
3219 end
3220 function luamplib.registergroup (boxid, name, opts)
3221   if opts.asgroup and opts.asgroup:find"wrapped" then
3222     luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3223     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
3224     opts.asgroup = opts.asgroup:gsub("wrapped","",)
3225   end
3226   local box = texgetbox(boxid)
3227   local wd, ht, dp = node.getwhd(box)

```

```

3228 local is_mask = opts.asgroup and opts.asgroup:find"masking"
3229 local res = opts.resources or ""
3230 res = gather_resources(res)
3231 local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3232 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
3233 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
3234 if opts.matrix and opts.matrix:find"%a" then
3235     local t = get_mp_matrix(opts.matrix)
3236     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
3237 end
3238 local grtype = 3
3239 if opts.bbox then
3240     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3241     grtype = 2
3242 end
3243 local mpllx, mplly = get_macro'MPlLx', get_macro'MPlly'
3244 if is_mask then
3245     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3246     t[5], t[6] = t[5]+mpllx, t[6]+mplly
3247     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
3248     mpllx, mplly = 0, 0
3249 end
3250 if opts.matrix then
3251     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3252     grtype = opts.bbox and 4 or 1
3253 end
3254 if opts.asgroup and not opts.asgroup:find"off" then
3255     local t = { isolated = false, knockout = false, masking = false }
3256     for _,v in ipairs(opts.asgroup:gsub("%s", ""):explode",+") do t[v] = true end
3257     local on
3258     if t.masking then
3259         on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorspace or "/DeviceGray"))
3260     else
3261         local cs = opts.colorspace and ("/CS%s"):format(opts.colorspace) or ""
3262         on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3263     end
3264     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3265 end
3266 local trgroup = pdfetcs.tr_group
3267 trgroup.shifts[name] = { mpllx, mplly }
3268 local whd
3269 if pdfmode then
3270     attr = tableconcat(attr) :gsub(decimals,rmzeros)
3271     local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3272     token.set_macro("luamplib.group"..name, tableconcat{
3273         "\\useboxresource ", index,
3274     }, "global")
3275     whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3276 else

```

```

3277   trgroup.cnt = (trgroup.cnt or 0) + 1
3278   local objname = format("@mplibtrgr%s", trgroup.cnt)
3279   texsprint {
3280     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3281     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3282     "\\hbox{\\unhbox ", boxid, "}}\\luamplibatnextshipout{",
3283     "\\special{pdf:bcontent}",
3284     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3285     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3286     "\\special{pdf:put @resources <<", res, ">>}",
3287     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3288     "\\special{pdf:econtent}}",
3289   }
3290   token.set_macro("luamplib.group"..name, tableconcat{
3291     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3292     "\\wd\\mplibscratchbox ", wd, "sp",
3293     "\\ht\\mplibscratchbox ", ht, "sp",
3294     "\\dp\\mplibscratchbox ", dp, "sp",
3295     "\\box\\mplibscratchbox",
3296   }, "global")
3297   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3298   end
3299   info("w/h/d of group '%s': %s", name, whd)
3300 end
3301

```

luamplib.convert: flushing figures

```

3302 do
3303   local function stop_special_effects(fade,opaq)
3304     if fade then -- fading
3305       stop_pdf_code()
3306     end
3307     if opaq then -- opacity
3308       pdf_literalcode(opaq)
3309     end
3310   end
3311

```

For parsing prescript materials.

```

3312   local function script2table(s)
3313     local t = {}
3314     for _,i in ipairs(s:explode("\\13+")) do
3315       local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3316       if k and v and k ~= "" and not t[k] then
3317         t[k] = v
3318       end
3319     end
3320     return t
3321   end
3322

```

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

```

3323 local function pdf_textfigure(font,size,text,width,height,depth)
3324   text = text:gsub(".",function(c)
3325     return format("\\hbox{\\char%i}",string.byte(c)) -- kerning happens in metapost : false
3326   end)
3327   put2output("\\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
3328 end
3329
3330 local bend_tolerance = 131/65536
3331
3332 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3333
3334 local function pen_characteristics(object)
3335   local t = mplib.pen_info(object)
3336   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3337   divider = sx*sy - rx*ry
3338   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3339 end
3340
3341 local function concat(px, py) -- no tx, ty here
3342   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
3343 end
3344
3345 local function curved(ith,pth)
3346   local d = pth.left_x - ith.right_x
3347   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3348     abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3349     d = pth.left_y - ith.right_y
3350     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3351       abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3352       return false
3353     end
3354   end
3355   return true
3356 end
3357
3358 local function flushnormalpath(path,open)
3359   local pth, ith
3360   for i=1,#path do
3361     pth = path[i]
3362     if not ith then
3363       pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3364     elseif curved(ith,pth) then
3365       pdf_literalcode("%f %f %f %f %f %f c",
3366         ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3367     else
3368       pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3369     end

```

```

3370     ith = pth
3371 end
3372 if not open then
3373     local one = path[1]
3374     if curved(pth,one) then
3375         pdf_literalcode("%f %f %f %f %f %f c",
3376             pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3377     else
3378         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3379     end
3380 elseif #path == 1 then -- special case .. draw point
3381     local one = path[1]
3382     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3383 end
3384 end
3385
3386 local function flushconcatpath(path,open)
3387     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3388     local pth, ith
3389     for i=1,#path do
3390         pth = path[i]
3391         if not ith then
3392             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3393         elseif curved(ith,pth) then
3394             local a, b = concat(ith.right_x,ith.right_y)
3395             local c, d = concat(pth.left_x,pth.left_y)
3396             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3397         else
3398             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3399         end
3400         ith = pth
3401     end
3402     if not open then
3403         local one = path[1]
3404         if curved(pth,one) then
3405             local a, b = concat(pth.right_x,pth.right_y)
3406             local c, d = concat(one.left_x,one.left_y)
3407             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3408         else
3409             pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3410         end
3411     elseif #path == 1 then -- special case .. draw point
3412         local one = path[1]
3413         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3414     end
3415 end
3416

```

Finally, flush figures by inserting PDF literals.

```

3417 local function flush (result,flusher)
3418   if result then
3419     local figures = result.fig
3420     if figures then
3421       for f=1, #figures do
3422         info("flushing figure %s",f)
3423         local figure = figures[f]
3424         local objects = figure:objects()
3425         local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3426         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3427         local bbox = figure:boundingbox()
3428         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3429         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()

3430   else

For legacy behavior, insert 'pre-fig' TEX code here.

3431   if tex_code_pre_mplib[f] then
3432     put2output(tex_code_pre_mplib[f])
3433   end
3434   pdf_startfigure(fignum,llx,lly,urx,ury)
3435   start_pdf_code()
3436   if objects then
3437     local savedpath = nil
3438     local savedhtap = nil
3439     for o=1,#objects do
3440       local object      = objects[o]
3441       local objecttype  = object.type

```

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

```

3442     local prescript      = object.prescript
3443     prescript = prescript and script2table(prescript) -- prescript is now a table
3444     do_preobj_CR(object,prescript) -- color
3445     local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3446     local fading_ = do_preobj_FADE(object,prescript) -- fading
3447     do_preobj_PAT(object,prescript) -- tiling pattern
3448     do_preobj_shading(object,prescript) -- shading pattern
3449     local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3450     if prescript and prescript.mplibtexboxid then
3451       put_tex_boxes(object,prescript)
3452     elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3453     elseif objecttype == "start_clip" then
3454       local evenodd = not object.istext and object.postscript == "evenodd"
3455       start_pdf_code()

```

```

3456         flushnormalpath(object.path,false)
3457         pdf_literalcode(evenodd and "W* n" or "W n")
3458     elseif objecttype == "stop_clip" then
3459         stop_pdf_code()
3460         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3461     elseif objecttype == "special" then

```

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

```

3462         if prescript and prescript.postmplibverbtx then
3463             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
3464         end
3465     elseif objecttype == "text" then
3466         local ot = object.transform -- 3,4,5,6,1,2
3467         start_pdf_code()
3468         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3469         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3470         stop_pdf_code()
3471     elseif not trgroup and fading_ ~= "stop" then
3472         local evenodd, collect, both = false, false, false
3473         local postscript = object.postscript
3474         if not object.istext then
3475             if postscript == "evenodd" then
3476                 evenodd = true
3477             elseif postscript == "collect" then
3478                 collect = true
3479             elseif postscript == "both" then
3480                 both = true
3481             elseif postscript == "eoboth" then
3482                 evenodd = true
3483                 both = true
3484             end
3485         end
3486         if collect then
3487             if not savedpath then
3488                 savedpath = { object.path or false }
3489                 savedhtap = { object.htap or false }
3490             else
3491                 savedpath[#savedpath+1] = object.path or false
3492                 savedhtap[#savedhtap+1] = object.htap or false
3493             end
3494         else

```

Removed from ConTeXt general: color stuff.

```

3495         local ml = object.miterlimit
3496         if ml and ml ~= miterlimit then
3497             miterlimit = ml
3498             pdf_literalcode("%f M",ml)
3499         end
3500         local lj = object.linejoin
3501         if lj and lj ~= linejoin then

```

```

3502         linejoin = lj
3503         pdf_literalcode("%i j",lj)
3504     end
3505     local lc = object.linecap
3506     if lc and lc ~= linecap then
3507         linecap = lc
3508         pdf_literalcode("%i J",lc)
3509     end
3510     local dl = object.dash
3511     if dl then
3512         local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3513         if d ~= dashed then
3514             dashed = d
3515             pdf_literalcode(dashed)
3516         end
3517     elseif dashed then
3518         pdf_literalcode("[ ] 0 d")
3519         dashed = false
3520     end
3521     local path = object.path
3522     local transformed, penwidth = false, 1
3523     local open = path and path[1].left_type and path[#path].right_type
3524     local pen = object.pen
3525     if pen then
3526         if pen.type == 'elliptical' then
3527             transformed, penwidth = pen_characteristics(object) -- boolean, value
3528             pdf_literalcode("%f w",penwidth)
3529             if objecttype == 'fill' then
3530                 objecttype = 'both'
3531             end
3532         else -- calculated by mplib itself
3533             objecttype = 'fill'
3534         end
3535     end
end

```

Added : shading

```

3536     local shade_no, shade_stroke, shade_cm = do_preobj_SH(object,prescript) -- shading
3537     if shade_no then
3538         pdf_literalcode"q /Pattern cs"
3539         objecttype = false
3540     end
3541     if transformed then
3542         start_pdf_code()
3543     end
3544     if path then
3545         if savedpath then
3546             for i=1,#savedpath do
3547                 local path = savedpath[i]
3548                 if transformed then

```

```

3549         flushconcatpath(path,open)
3550     else
3551         flushnormalpath(path,open)
3552     end
3553 end
3554 savedpath = nil
3555 end
3556 if transformed then
3557     flushconcatpath(path,open)
3558 else
3559     flushnormalpath(path,open)
3560 end
3561 if objecttype == "fill" then
3562     pdf_literalcode(evenodd and "h f*" or "h f")
3563 elseif objecttype == "outline" then
3564     if both then
3565         pdf_literalcode(evenodd and "h B*" or "h B")
3566     else
3567         pdf_literalcode(open and "S" or "h S")
3568     end
3569 elseif objecttype == "both" then
3570     pdf_literalcode(evenodd and "h B*" or "h B")
3571 end
3572 end
3573 if transformed then
3574     stop_pdf_code()
3575 end
3576 local path = object.htap

```

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

```

3577 if path then
3578     if transformed then
3579         start_pdf_code()
3580     end
3581     if savedhtap then
3582         for i=1,#savedhtap do
3583             local path = savedhtap[i]
3584             if transformed then
3585                 flushconcatpath(path,open)
3586             else
3587                 flushnormalpath(path,open)
3588             end
3589         end
3590         savedhtap = nil
3591         evenodd = true
3592     end
3593     if transformed then
3594         flushconcatpath(path,open)
3595     else

```

```

3596         flushnormalpath(path,open)
3597     end
3598     if objecttype == "fill" then
3599         pdf_literalcode(evenodd and "h f*" or "h f")
3600     elseif objecttype == "outline" then
3601         pdf_literalcode(open and "S" or "h S")
3602     elseif objecttype == "both" then
3603         pdf_literalcode(evenodd and "h B*" or "h B")
3604     end
3605     if transformed then
3606         stop_pdf_code()
3607     end
3608 end

```

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

```

3609         if shade_no then -- shading
3610             pdf_literalcode("W%s %s %s/MPLibSh%s sh Q",
3611                 evenodd and "*" or "",
3612                 shade_stroke and "s" or "n",
3613                 shade_cm and shade_cm.." cm " or "",
3614                 shade_no)
3615         end
3616     end
3617 end
3618 if fading_ == "start" then
3619     pdfetcs.fading.specialeffects = {fading_, tr_opaq}
3620 elseif trgroup == "start" then
3621     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq}
3622 elseif fading_ == "stop" then
3623     local se = pdfetcs.fading.specialeffects
3624     if se then stop_special_effects(se[1], se[2]) end
3625 elseif trgroup == "stop" then
3626     local se = pdfetcs.tr_group.specialeffects
3627     if se then stop_special_effects(se[1], se[2]) end
3628 else
3629     stop_special_effects(fading_, tr_opaq)
3630 end
3631 if fading_ or trgroup then -- extgs resetted
3632     miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3633 end
3634 end
3635 end
3636 stop_pdf_code()
3637 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimt_Ex code.

```

3638 for _,v in ipairs(figcontents) do
3639     if type(v) == "table" then
3640         textsprint"\mplibtoPDF{"; textsprint(v[1], v[2]); textsprint"}"
3641     else

```

```

3642         texsprint(v)
3643     end
3644 end
3645     if #figcontents.post > 0 then texsprint(figcontents.post) end
3646     figcontents = { post = { } }
3647 end
3648 end
3649 end
3650 end
3651 end
3652
3653 function luamplib.convert (result, flusher)
3654     flush(result, flusher)
3655     return true -- done
3656 end
3657 end
3658
3659 function luamplib.colorconverter (cr)
3660     local n = #cr
3661     if n == 4 then
3662         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3663         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K", c,m,y,k,c,m,y,k), "0 g 0 G"
3664     elseif n == 3 then
3665         local r, g, b = cr[1], cr[2], cr[3]
3666         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG", r,g,b,r,g,b), "0 g 0 G"
3667     else
3668         local s = cr[1]
3669         return format("%.3f g %.3f G", s,s), "0 g 0 G"
3670     end
3671 end

```

2.2 T_EX package

First we need to load some packages.

```

3672 \ifcsname ProvidesPackage\endcsname

```

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

```

3673 \NeedsTeXFormat{LaTeX2e}
3674 \ProvidesPackage{luamplib}
3675 [2026/08/07 v2.42.5 mplib package for LuaTeX]
3676 \fi
3677 \ifdefined\newluafunction\else
3678 \input ltluatex
3679 \fi

```

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

```

3680 \ifnum\outputmode=0
3681   \ifdefined\AddToHookNext
3682     \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3683     \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3684     \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3685   \else
3686     \input atbegshi.sty
3687     \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3688     \let\luamplibatfirstshipout\AtBeginShipoutFirst
3689     \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3690   \fi
3691 \fi

```

Loading of lua code.

```

3692 \directlua{require("luamplib")}

```

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

```

3693 \ifx\pdfoutput\undefined
3694   \let\pdfoutput\outputmode
3695 \fi
3696 \ifx\pdfliteral\undefined
3697   \protected\def\pdfliteral{\pdfextension literal}
3698 \fi

```

Set the format for METAPOST.

```

3699 \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.

```

3700 \ifnum\pdfoutput>0
3701   \let\mplibtoPDF\pdfliteral
3702 \else
3703   \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3704   \ifcsname PackageInfo\endcsname
3705     \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3706   \else
3707     \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3708   \fi
3709 \fi

```

To make mplibcode typeset always in horizontal mode.

```

3710 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3711 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3712 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

3713 \def\mplibsetupcatcodes{%
3714   %catcode`\{=12 %catcode`\}=12
3715   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3716   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^M=12
3717 }

```

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

```

3718 \ifnum\outputmode>0 %
3719   \def\mplibputtextbox#1#2#3#4{%
3720     \pdfextension save\relax
3721     \vbox to 0pt{\vss
3722       \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
3723       \kern#3bp}%
3724     \pdfextension restore\relax}
3725 \else
3726   \def\mplibputtextbox#1#2#3#4{%
3727     \special{pdf:btrans matrix #4 #2 #3}%
3728     \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
3729     \special{pdf:etrans}}
3730 \fi

```

use Transparency Group

```

3731 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3732 \def\usemplibgroupmain#1{%
3733   \prependtomplibox\hbox dir TLT\bgroup
3734   \csname luamplib.group.#1\endcsname
3735   \egroup
3736 }
3737 \protected\def\mplibgroup#1{%
3738   \begingroup
3739   \def\MPllx{0}\def\MPlly{0}%
3740   \def\mplibgroupname{#1}%
3741   \mplibgroupgetnexttok
3742 }
3743 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3744 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
3745 \def\mplibgroupbranch{%
3746   \ifx [\nexttok
3747     \expandafter\mplibgroupopts
3748   \else
3749     \ifx\mplibsptoken\nexttok
3750       \expandafter\expandafter\expandafter\mplibgroupskipspace
3751     \else
3752       \let\mplibgroupoptions\empty
3753       \expandafter\expandafter\expandafter\mplibgroupmain
3754     \fi
3755   \fi
3756 }
3757 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3758 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3759 \protected\def\endmplibgroup{\egroup
3760   \directlua{ luamplib.registergroup(
3761     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3762   )}%

```

```

3763 \endgroup
3764 }

Patterns

3765 \def\:\global\let\mplibsptoken= } \: }
3766 \protected\def\mppattern#1{%
3767 \begingroup
3768 \def\MPllx{0}\def\MPlly{0}%
3769 \def\mplibpatternname{#1}%
3770 \mplibpatterngetnexttok
3771 }
3772 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3773 \def\mplibpatternskipspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3774 \def\mplibpatternbranch{%
3775 \ifx [\nexttok
3776 \expandafter\mplibpatternopts
3777 \else
3778 \ifx\mplibsptoken\nexttok
3779 \expandafter\expandafter\expandafter\mplibpatternskipspace
3780 \else
3781 \let\mplibpatternoptions\empty
3782 \expandafter\expandafter\expandafter\mplibpatternmain
3783 \fi
3784 \fi
3785 }
3786 \def\mplibpatternopts[#1]{%
3787 \def\mplibpatternoptions{#1}%
3788 \mplibpatternmain
3789 }
3790 \def\mplibpatternmain{%
3791 \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3792 }
3793 \protected\def\endmppattern{%
3794 \egroup
3795 \directlua{ luamplib.registerpattern(
3796 \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3797 )}%
3798 \endgroup
3799 }

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

3800 \def\mpfiginstancename{@mpfig}
3801 \protected\def\mpfig{%
3802 \begingroup
3803 \futurelet\nexttok\mplibmpfigbranch
3804 }
3805 \def\mplibmpfigbranch{%
3806 \ifx *\nexttok
3807 \expandafter\mplibprempfig
3808 \else

```

```

3809 \ifx [\nexttok
3810 \expandafter\expandafter\expandafter\mplibgobbleoptsmpfig
3811 \else
3812 \expandafter\expandafter\expandafter\mplibmainmpfig
3813 \fi
3814 \fi
3815 }
3816 \def\mplibgobbleoptsmpfig[#1]{\mplibmainmpfig}
3817 \def\mplibmainmpfig{%
3818 \begingroup
3819 \mplibsetupcatcodes
3820 \mplibdomainmpfig
3821 }
3822 \long\def\mplibdomainmpfig#1\endmpfig{%
3823 \endgroup
3824 \directlua{
3825 local legacy = luamplib.legacyverbatim
3826 local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3827 local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3828 luamplib.legacyverbatim = false
3829 luamplib.everymplib["\mpfiginstancename"] = ""
3830 luamplib.everyendmplib["\mpfiginstancename"] = ""
3831 luamplib.process_mplibcode(
3832 "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===].." "..everyendmpfig.." endfig;",
3833 "\mpfiginstancename")
3834 luamplib.legacyverbatim = legacy
3835 luamplib.everymplib["\mpfiginstancename"] = everympfig
3836 luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3837 }%
3838 \endgroup
3839 }
3840 \def\mplibprempfig#1{%
3841 \begingroup
3842 \mplibsetupcatcodes
3843 \mplibdoprempfig
3844 }
3845 \long\def\mplibdoprempfig#1\endmpfig{%
3846 \endgroup
3847 \directlua{
3848 local legacy = luamplib.legacyverbatim
3849 local everympfig = luamplib.everymplib["\mpfiginstancename"]
3850 local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3851 luamplib.legacyverbatim = false
3852 luamplib.everymplib["\mpfiginstancename"] = ""
3853 luamplib.everyendmplib["\mpfiginstancename"] = ""
3854 luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
3855 luamplib.legacyverbatim = legacy
3856 luamplib.everymplib["\mpfiginstancename"] = everympfig
3857 luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig

```

```

3858 }%
3859 \endgroup
3860 }
3861 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3862 \unless\ifcsname ver@luamplib.sty\endcsname
3863 \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3864 \protected\def\mplibcode{%
3865 \begingroup
3866 \futurelet\nexttok\mplibcodebranch
3867 }
3868 \def\mplibcodebranch{%
3869 \ifx [\nexttok
3870 \expandafter\mplibcodegetinstancename
3871 \else
3872 \global\let\currentmpinstancename\empty
3873 \expandafter\mplibcodeindeed
3874 \fi
3875 }
3876 \def\mplibcodeindeed{%
3877 \begingroup
3878 \mplibsetupcatcodes
3879 \mplibdocode
3880 }
3881 \long\def\mplibdocode#1\endmplibcode{%
3882 \endgroup
3883 \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]==],"\currentmpinstancename")}%
3884 \endgroup
3885 }
3886 \protected\def\endmplibcode{endmplibcode}
3887 \else

```

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

```

3888 \newenvironment{mplibcode}[1][{}]{%
3889 \xdef\currentmpinstancename{#1}%
3890 \mplibtmptoks{}\ltxdomplibcode
3891 }{}
3892 \def\ltxdomplibcode{%
3893 \begingroup
3894 \mplibsetupcatcodes
3895 \ltxdomplibcodeindeed
3896 }
3897 \def\mplib@mplibcode{mplibcode}
3898 \long\def\ltxdomplibcodeindeed#1\end#2{%
3899 \endgroup
3900 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3901 \def\mplibtemp@a{#2}%
3902 \ifx\mplib@mplibcode\mplibtemp@a
3903 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]==],"\currentmpinstancename")}%

```

```

3904     \end{mplibcode}%
3905     \else
3906         \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3907         \expandafter\ltxdomplibcode
3908     \fi
3909 }
3910 \fi

    User settings.
3911 \def\mplibshowlog#1{\directlua{
3912     local s = string.lower("#1")
3913     if s == "enable" or s == "true" or s == "yes" then
3914         luampLib.showlog = true
3915     else
3916         luampLib.showlog = false
3917     end
3918 }}
3919 \def\mpliblegacybehavior#1{\directlua{
3920     local s = string.lower("#1")
3921     if s == "enable" or s == "true" or s == "yes" then
3922         luampLib.legacyverbatim = true
3923     else
3924         luampLib.legacyverbatim = false
3925     end
3926 }}
3927 \def\mplibverbatim#1{\directlua{
3928     local s = string.lower("#1")
3929     if s == "enable" or s == "true" or s == "yes" then
3930         luampLib.verbatiminput = true
3931     else
3932         luampLib.verbatiminput = false
3933     end
3934 }}
3935 \newtoks\mplibtmptoks

    \everymplib & \everyendmplib: macros resetting luampLib.every(end)mpLib tables
3936 \ifcsname ver@luampLib.sty\endcsname
3937     \protected\def\everymplib{%
3938         \begingroup
3939         \mplibsetupcatcodes
3940         \mplibdoeverymplib
3941     }
3942     \protected\def\everyendmplib{%
3943         \begingroup
3944         \mplibsetupcatcodes
3945         \mplibdoeveryendmplib
3946     }
3947     \newcommand\mplibdoeverymplib[2][{}]{%
3948         \endgroup
3949         \directlua{

```

```

3950     luamplib.everymplib["#1"] = [===[\unexpanded{#2}]==]
3951   }%
3952 }
3953 \newcommand\mplibdoeveryendmplib[2][{}]{%
3954   \endgroup
3955   \directlua{
3956     luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]==]
3957   }%
3958 }
3959 \else
3960   \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3961   \protected\def\everymplib#1#{%
3962     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3963     \begingroup
3964     \mplibsetupcatcodes
3965     \mplibdoeverymplib
3966   }
3967   \long\def\mplibdoeverymplib#1{%
3968     \endgroup
3969     \directlua{
3970       luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]==]
3971     }%
3972   }
3973   \protected\def\everyendmplib#1#{%
3974     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3975     \begingroup
3976     \mplibsetupcatcodes
3977     \mplibdoeveryendmplib
3978   }
3979   \long\def\mplibdoeveryendmplib#1{%
3980     \endgroup
3981     \directlua{
3982       luamplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]==]
3983     }%
3984   }
3985 \fi

```

TeX macros for dimen/color

```

3986 \def\mpdim#1{ \runscript("luamplibdimen{#1}") }
3987 \def\mpcolor#1#{\domplibcolor{#1}}
3988 \def\domplibcolor#1#2{ \runscript("luamplibcolor{#1{#2}}") }

```

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

```

3989 \def\mplibnumbersystem#1{\directlua{
3990   local t = "#1"
3991   if t == "binary" then t = "decimal" end
3992   luamplib.numbersystem = t
3993 }}

```

Settings for .mp cache files.

```

3994 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
3995 \def\mplibdomakenocache#1,{%
3996   \ifx\empty#1\empty
3997     \expandafter\mplibdomakenocache
3998   \else
3999     \ifx\stop#1\else
4000       \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
4001     \expandafter\expandafter\expandafter\mplibdomakenocache
4002     \fi
4003   \fi
4004 }
4005 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4006 \def\mplibdocancelnocache#1,{%
4007   \ifx\empty#1\empty
4008     \expandafter\mplibdocancelnocache
4009   \else
4010     \ifx\stop#1\else
4011       \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
4012     \expandafter\expandafter\expandafter\mplibdocancelnocache
4013     \fi
4014   \fi
4015 }
4016 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4017 \def\mplibtexttextlabel#1{\directlua{
4018   local s = string.lower("#1")
4019   if s == "enable" or s == "true" or s == "yes" then
4020     luamplib.texttextlabel = true
4021   else
4022     luamplib.texttextlabel = false
4023   end
4024 }}
4025 \def\mplibcodeinherit#1{\directlua{
4026   local s = string.lower("#1")
4027   if s == "enable" or s == "true" or s == "yes" then
4028     luamplib.codeinherit = true
4029   else
4030     luamplib.codeinherit = false
4031   end
4032 }}
4033 \def\mplibglobaltexttext#1{\directlua{
4034   local s = string.lower("#1")
4035   if s == "enable" or s == "true" or s == "yes" then
4036     luamplib.globaltexttext = true
4037   else
4038     luamplib.globaltexttext = false
4039   end
4040 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```
4041 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi
```

We encapsulate the literals.

```
4042 \def\mplibstarttoPDF#1#2#3#4{%
4043   \prependtomplibbox
4044   \hbox dir TLT\bgroup
4045   \xdef\MPllx{#1}\xdef\MPlly{#2}%
4046   \xdef\MPurx{#3}\xdef\MPury{#4}%
4047   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4048   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4049   \parskip0pt%
4050   \leftskip0pt%
4051   \parindent0pt%
4052   \everypar{}%
4053   \setbox\mplibscratchbox\ vbox\bgroup
4054   \noindent
4055 }
4056 \def\mplibstoptoPDF{%
4057   \par
4058   \egroup %
4059   \setbox\mplibscratchbox\ hbox %
4060     {\hskip-\MPllx bp%
4061      \raise-\MPlly bp%
4062      \box\mplibscratchbox}%
4063   \setbox\mplibscratchbox\ vbox to \MPheight
4064     {\vfill
4065      \hsize\MPwidth
4066      \wd\mplibscratchbox0pt%
4067      \ht\mplibscratchbox0pt%
4068      \dp\mplibscratchbox0pt%
4069      \box\mplibscratchbox}%
4070   \wd\mplibscratchbox\MPwidth
4071   \ht\mplibscratchbox\MPheight
4072   \box\mplibscratchbox
4073   \egroup
4074 }
```

Text items have a special handler.

```
4075 \def\mplibtexttext#1#2#3#4#5{%
4076   \begingroup
4077   \setbox\mplibscratchbox\ hbox
4078     {\font\temp=#1 at #2bp%
4079      \temp
4080      #3}%
4081   \setbox\mplibscratchbox\ hbox
4082     {\hskip#4 bp%
4083      \raise#5 bp%
4084      \box\mplibscratchbox}%

```

```

4085 \wd\mplibscratchbox0pt%
4086 \ht\mplibscratchbox0pt%
4087 \dp\mplibscratchbox0pt%
4088 \box\mplibscratchbox
4089 \endgroup
4090 }

```

Input luamplib.cfg when it exists.

```

4091 \openin0=luamplib.cfg
4092 \ifeof0 \else
4093 \closein0
4094 \input luamplib.cfg
4095 \fi

```

Code for tagpdf

```

4096 \def\luamplibtagtextboxset#1#2{#2}
4097 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4098 \let\luamplibtagasgroupset\relax
4099 \let\luamplibtagasgroupput\luamplibtagtextboxset
4100 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4101 \ifcsname ver@tagpdf.sty\endcsname \else
4102 \ExplSyntaxOn
4103 \keys_define:nn{luamplib/tagging}
4104 {
4105 ,alt .code:n = { }
4106 ,actualtext .code:n = { }
4107 ,artifact .code:n = { }
4108 ,text .code:n = { }
4109 ,off .code:n = { }
4110 ,tag .code:n = { }
4111 ,adjust-BBox .code:n = { }
4112 ,tagging-setup .code:n = { }
4113 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4114 ,instancename .meta:n = { instance = {#1} }
4115 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4116 }
4117 \RenewDocumentCommand\mplibcode{0{}}
4118 {
4119 \tl_gclear:N \currentmpinstancename
4120 \keys_set:ne{luamplib/tagging}{#1}
4121 \mplibtmptoks{}\ltxdomplibcode
4122 }
4123 \cs_set_eq:NN \mplibaltext \use_none:n
4124 \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.

```

4125 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4126 \ExplSyntaxOff
4127 \endinput\fi
4128 \ExplSyntaxOn
4129 \tl_new:N \l__luamplib_tag_envname_tl
4130 \tl_new:N \l__luamplib_tag_alt_tl
4131 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4132 \tl_new:N \l__luamplib_tag_actual_tl
4133 \tl_new:N \l__luamplib_tag_struct_tl
4134 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4135 \bool_new:N \l__luamplib_tag_usetext_bool
4136 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4137 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4138 \tl_new:N \l__luamplib_tag_bbox_draw_tl
4139 \tl_new:N \l__luamplib_BBoxllx_tl
4140 \tl_new:N \l__luamplib_BBoxlly_tl
4141 \tl_new:N \l__luamplib_BBoxurx_tl
4142 \tl_new:N \l__luamplib_BBoxury_tl
4143 \msg_new:nnn {luamplib}{figure-text-reuse}
4144 {
4145   tex-text~box~#1~probably~is~incorrectly~tagged.~
4146   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4147   Check~the~resulting~PDF.
4148 }
4149 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4150 {
4151   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4152   Using~mplibgroup~with~text~mode~is~not~recommended.~
4153   Check~the~resulting~PDF.
4154 }
4155 \msg_new:nnn {luamplib}{alt-text-missing}
4156 {
4157   Alternate~text~for~#1~is~missing.~
4158   Using~the~default~value~'#2'~instead.
4159 }

```

Sockets for tex-text boxes.

```

4160 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4161 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4162 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4163 {

```

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.

```

4164 \bool_if:NTF \l__luamplib_tag_usetext_bool
4165 {
4166   \tag_mc_end_push:
4167   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4168   \cs_gset_nopar:cpe {luamplib.taggedbox.#1} {\tag_get:n{struct_num}}

```

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

```

4169 \tag_mc_begin:n{tag=text}
4170 #2
4171 \tag_mc_end:
4172 \tag_struct_end:
4173 \tag_mc_begin_pop:n{ }
4174 }
4175 {
4176 \tag_suspend:n{\luamplibtagtextboxset}
4177 #2
4178 \tag_resume:n{\luamplibtagtextboxset}
4179 }
4180 }
4181 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4182 {
4183 \bool_lazy_and:nnTF
4184 { \l__luamplib_tag_usetext_bool }
4185 { \cs_if_free_p:c {luamplib.nottaggedbox.#1} }
4186 {
4187 \tag_resume:n{\mplibputtextbox}
4188 \tag_mc_end:
4189 \cs_if_exist:cTF {luamplib.taggedbox.#1}
4190 {
4191 \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4192 #2
4193 \cs_undefine:c {luamplib.taggedbox.#1}
4194 }
4195 {
4196 \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
4197 \tag_mc_begin:n{ }
4198 \int_set:Nn \l_tmpa_int {#1}
4199 \tag_mc_reset_box:N \l_tmpa_int
4200 #2
4201 \tag_mc_end:
4202 }
4203 \tag_mc_begin:n{artifact}
4204 }
4205 {
4206 \int_set:Nn \l_tmpa_int {#1}
4207 \tag_mc_reset_box:N \l_tmpa_int
4208 #2
4209 }
4210 }
4211 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4212 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4213 \cs_set_nopar:Npn \luamplibtagtextboxset
4214 {
4215 \tag_socket_use:nnn{luamplib/texttext/set}
4216 }

```

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

```

4217 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4218 {
4219   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
4220   \bool_set_false:N \l__luamplib_tag_usetext_bool
4221   \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
4222   \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4223   \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4224 }
4225 \sys_if_output_pdf:TF
4226 {
4227   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4228   {
4229     \pdfextension save\relax
4230     \vbox to 0pt{\vss
4231       \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4232         \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4233       \kern #3bp}
4234     \pdfextension restore\relax
4235   }
4236 }
4237 {
4238   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4239   {
4240     \special{pdf:btrans~matrix~#4~#2~#3}
4241     \vbox to 0pt{\vss\hbox bdir0 to 0pt{
4242       \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4243     \special{pdf:etrans}
4244   }
4245 }

```

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

```

4246 \cs_set_nopar:Npn \luamplibtagasgroupset
4247 {
4248   \bool_set_false:N \l__luamplib_tag_usetext_bool
4249 }
4250 \cs_set_nopar:Npn \luamplibtagasgroupput
4251 {
4252   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4253   \tag_socket_use:nnn{luamplib/mplibgroup/put}
4254 }

```

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

```

4255 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4256 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4257 {
4258   \cs_if_free:cT {luamplib.mplibgroup.text.#1}

```

```

4259 {
4260   \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4261   \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4262 }
4263 \tag_mc_end:
4264 \tag_mc_begin:n{tag=text}
4265 #2
4266 \tag_mc_end:
4267 \tag_mc_begin:n{artifact}
4268 }
4269 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

A macro for BBox attribute

```

4270 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4271 {
4272   \tl_set:Nc \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4273   \tex_savepos:D
4274   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4275   \tl_set:Nc \l__luamplib_BBox_llx_tl
4276     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4277   \tl_set:Nc \l__luamplib_BBox_lly_tl
4278     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4279   \tl_set:Nc \l__luamplib_BBox_urx_tl
4280     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4281   \tl_set:Nc \l__luamplib_BBox_ury_tl
4282     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4283   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4284   {
4285     \int_zero:N \l_tmpa_int
4286     \tl_map_inline:nn
4287     {
4288       \l__luamplib_BBox_llx_tl
4289       \l__luamplib_BBox_lly_tl
4290       \l__luamplib_BBox_urx_tl
4291       \l__luamplib_BBox_ury_tl
4292     }
4293     {
4294       \int_incr:N \l_tmpa_int
4295       \tl_set:Nc ##1
4296       {
4297         \fp_eval:n
4298         {
4299           ##1
4300           +
4301           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4302         }
4303       }
4304     }
4305   }

```

```

4306 \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4307 {
4308   /O /Layout /BBox [
4309     \l__luamplib_BBox_llx_tl\c_space_tl
4310     \l__luamplib_BBox_lly_tl\c_space_tl
4311     \l__luamplib_BBox_urx_tl\c_space_tl
4312     \l__luamplib_BBox_ury_tl
4313   ]
4314 }
4315 \bool_if:NT \l__tag_graphic_debug_bool
4316 {
4317   \iow_log:e
4318   {
4319     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4320     \l__luamplib_BBox_llx_tl\c_space_tl
4321     \l__luamplib_BBox_lly_tl\c_space_tl
4322     \l__luamplib_BBox_urx_tl\c_space_tl
4323     \l__luamplib_BBox_ury_tl
4324   }
4325   \sys_if_output_pdf:TF
4326   {
4327     \tl_set:Nc \l__luamplib_tag_bbox_draw_tl
4328     {
4329       \pdfextension save\relax
4330       \opacity_select:n{0.5} \color_select:n{red}
4331       \pdfextension literal~text
4332       {
4333         \l__luamplib_BBox_llx_tl\c_space_tl
4334         \l__luamplib_BBox_lly_tl\c_space_tl
4335         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4336         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4337         re~f
4338       }
4339       \pdfextension restore\relax
4340     }
4341   }
4342   {
4343     \tl_set:Nc \l__luamplib_tag_bbox_draw_tl
4344     {
4345       \special{pdf:bcontent}
4346       \opacity_select:n{0.5} \color_select:n{red}
4347       \special{pdf:code~
4348         1~0~0~1~
4349         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4350         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4351         cm
4352       }
4353       \special{pdf:code~
4354         \l__luamplib_BBox_llx_tl\c_space_tl

```

```

4355     \l__luamplib_BBox_lly_tl\c_space_tl
4356     \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4357     \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4358     re~f
4359   }
4360   \special{pdf:econtent}
4361 }
4362 }
4363 }
4364 }

```

Sockets for main process

```

4365 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4366 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4367 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4368 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4369 {
4370   \tag_mc_end_push:
4371   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4372   {
4373     \tl_if_empty:eTF{#1}
4374     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4375     { \tl_set:Ne \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4376     \msg_warning:nnVV{luamplib}{alt-text-missing}
4377     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4378   }
4379   \tag_struct_begin:n
4380   {
4381     tag=\l__luamplib_tag_struct_tl,
4382     alt=\l__luamplib_tag_alt_tl,
4383   }
4384   \tag_mc_begin:n{}
4385 }
4386 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4387 {
4388   \__luamplib_tag_bbox_attribute:n {#1}
4389   #2
4390   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4391   \tag_mc_end:
4392   \tag_struct_end:
4393   \tag_mc_begin_pop:n{}
4394 }
4395 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4396 {
4397   \tag_mc_end_push:
4398   \tag_struct_begin:n
4399   {
4400     tag=Span,
4401     actualtext=\l__luamplib_tag_actual_tl,

```

```

4402     }
4403     \tag_mc_begin:n{ }
4404 }
4405 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4406 {
4407     #2
4408     \tag_mc_end:
4409     \tag_struct_end:
4410     \tag_mc_begin_pop:n{ }
4411 }
4412 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4413 {
4414     \tag_mc_end_push:
4415     \tag_mc_begin:n{artifact}
4416 }
4417 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4418 {
4419     #2
4420     \tag_mc_end:
4421     \tag_mc_begin_pop:n{ }
4422 }

```

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

```

4423 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4424 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4425 {
4426     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4427     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4428 }
4429 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4430 {
4431     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4432     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4433 \prependtomplibbox \mplibnoforcehmode
4434 \mode_if_vertical:T { \noindent \aftergroup\par }
4435 }
4436 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4437 {
4438     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4439     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4440 }
4441 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4442 {
4443     \bool_set_true:N \l__luamplib_tag_usetext_bool
4444     \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4445     \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4446 \prependtomplibbox \mplibnoforcehmode

```

```

4447 \mode_if_vertical:T { \noindent \aftergroup\par }
4448 }
4449 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4450 {
4451   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4452   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4453 }
4454 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```

4455 \keys_define:nn{luamplib/tagging}
4456 {
4457   ,alt .code:n =
4458   {
4459     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4460     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4461   }
4462   ,actualtext .code:n =
4463   {
4464     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4465     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4466   }
4467   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4468   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4469   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4470   ,tag .code:n =
4471   {
4472     \str_case:nnF {#1}
4473     {
4474       {false} { \keys_set:nn {luamplib/tagging} {off} }
4475       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4476     }
4477     {
4478       \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4479       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4480     }
4481   }
4482   ,adjust-BBox .code:n =
4483   {
4484     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4485     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4486   }
4487   ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4488 }
4489 \keys_define:nn {luamplib/instance}
4490 {
4491   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4492   ,instancename .meta:n = { instance = {#1} }
4493   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }

```

4494 }

Redefine our macros

```
4495 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4496 {
4497   \prependtomplibbox
4498   \hbox_dir~TLT\bgroup
4499     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4500     \xdef\MPllyx{#1}\xdef\MPlly{#2}%
4501     \xdef\MPurx{#3}\xdef\MPury{#4}%
4502     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4503     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4504     \parskip0pt
4505     \leftskip0pt
4506     \parindent0pt
4507     \everypar{}%
4508     \setbox\mplibscratchbox\vbox\bgroup
4509       \tag_suspend:n{\mplibstarttoPDF}
4510       \noindent
4511 }
4512 \cs_set_nopar:Npn \mplibstoptoPDF
4513 {
4514   \par
4515   \egroup
4516   \setbox\mplibscratchbox\hbox
4517     {\hskip-\MPllx bp
4518      \raise-\MPlly bp
4519      \box\mplibscratchbox}%
4520   \setbox\mplibscratchbox\vbox to \MPheight
4521     {\vfill
4522      \hsize\MPwidth
4523      \wd\mplibscratchbox0pt
4524      \ht\mplibscratchbox0pt
4525      \dp\mplibscratchbox0pt
4526      \box\mplibscratchbox}%
4527   \wd\mplibscratchbox\MPwidth
4528   \ht\mplibscratchbox\MPheight
4529   \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4530   \egroup
4531 }
4532 \RenewDocumentCommand\mplibcode{0{}}
4533 {
4534   \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4535   \tl_gclear:N \currentmpinstancename
4536   \keys_set:known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4537   \keys_set:nV {luamplib/instance} \l_tmpa_tl
4538   \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4539   \tag_socket_use:n{luamplib/figure/init}
4540   \mplibtmptoks{}\ltxdomplibcode
```

```

4541 }
4542 \RenewDocumentCommand\mpfig{s 0{}}
4543 {
4544   \begingroup
4545   \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4546   \keys_set_known:ne {luamplib/tagging} {#2}
4547   \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4548   \tag_socket_use:n{luamplib/figure/init}
4549   \IfBooleanTF{#1} { \mplibprempfig * }
4550                   { \mplibmainmpfig }
4551 }
4552 \RenewDocumentCommand\usemplibgroup{0{ } m}
4553 {
4554   \begingroup
4555   \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4556   \keys_set_known:ne {luamplib/tagging} {#1}
4557   \tag_socket_use:n{luamplib/figure/init}
4558   \prependtomplibbox\hbox dir~TLT\bgroup
4559     \tag_socket_use:nn{luamplib/figure/begin}{#2}
4560     \setbox\mplibscratchbox\hbox\bgroup
4561       \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
4562       \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4563       \egroup
4564       \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4565     \egroup
4566   \endgroup
4567 }

```

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

```

4568 \cs_new_nopar:Npn \mplibalttext #1
4569 {
4570   \tl_set:Ne \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4571 }
4572 \cs_new_nopar:Npn \mplibactualtext #1
4573 {
4574   \tl_set:Ne \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4575 }
4576 \ExplSyntaxOff

```

That's all folks!

3 The GNU GPL License v2

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

GNU GENERAL PUBLIC LICENSE Version 2, June 1991 Copyright © 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.	you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program. In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.	Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
Preamble The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things. To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it. For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software. Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations. Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all. The precise terms and conditions for copying, distribution and modification follow.	4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following: (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or, (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.) The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or object form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable. If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.	NO WARRANTY 12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance. 6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it. 7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License. 8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program. If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances. It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a license cannot impose that choice. This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License. 9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License. 10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.	END OF TERMS AND CONDITIONS Appendix: How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. one line to give the program's name and a brief idea of what it does. Copyright (C) yyyy name of author This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA. Also add information on how to contact you by electronic and paper mail. If the program is interactive, make it output a short notice like this when it starts in an interactive mode: Gnomovision version 69, Copyright (C) yyyy name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details. The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program. You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names: Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker. signature of Ty Coon, 1 April 1989 Ty Coon, President of Vice This General Public License does not permit incorporating your program into proprietary programs. If your program is a subcomponent library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.