

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/08/12 V2.42.7

Abstract

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

Contents

1	Documentation	2
1.1	$\TeX$	3
1.1.1	Forcing horizontal mode	3
1.1.2	Insert some code into every code chunk	3
1.1.3	Choosing a METAPOST format	3
1.1.4	Choosing a number system	4
1.1.5	Showing METAPOST log	4
1.1.6	verbatimtex Legacy behavior	5
1.1.7	$\TeX$ -text labels	5
1.1.8	Inherit METAPOST code	6
1.1.9	<code>\mplibglobaltexttext</code>	6
1.1.10	Separate METAPOST instances	6
1.1.11	Verbatim input	7
1.1.12	$\TeX$ dimen	7
1.1.13	$\TeX$ color	7
1.1.14	<code>\mpfig</code> , <code>\endmpfig</code>	8
1.1.15	About cache files	9
1.1.16	About figure box metric	9
1.1.17	<code>luamplib.cfg</code>	9
1.1.18	Tagged PDF	9
1.2	METAPOST	11
1.2.1	$\TeX$ dimen and color	11
1.2.2	More on $\TeX$ color	11
1.2.3	Fill and stroke colors	12
1.2.4	Transparency	12

1.2.5	Fill and stroke transparency	13
1.2.6	Text outline as a text image	13
1.2.7	Glyph outline	14
1.2.8	Drawing a glyph outline	14
1.2.9	Text outline as a path image	15
1.2.10	Unicode-aware length	15
1.2.11	Unicode-aware substring	15
1.2.12	Shading	16
1.2.13	Fading	18
1.2.14	Tiling pattern	19
1.2.15	Form XObject from METAPOST side	22
1.2.16	Form XObject from T _E X side	25
1.2.17	Masking	26
1.2.18	Free-form Gouraud-shaded triangle mesh shading	27
1.2.19	Lattice-form Gouraud-shaded triangle mesh shading	28
1.2.20	Coons patch mesh shading	29
1.2.21	Tensor-product patch mesh shading	31
1.3	Lua	32
1.3.1	runscript	32
1.3.2	Accessing METAPOST variables	33
1.3.3	Running a METAPOST code	33
1.3.4	Registering a tiling pattern	33
1.3.5	Registering a form XObject	34
2	Implementation	34
2.1	Lua module	34
2.2	T _E Xpackage	115
3	The GNU GPL License v2	136

1 Documentation

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

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

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

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

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

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

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

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

1.1 $\TeX$

1.1.1 Forcing horizontal mode

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

1.1.2 Insert some code into every code chunk

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

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



1.1.3 Choosing a METAPOST format

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

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

N.B. As *metafun* is such a complicated format, we cannot support all the special effects provided by *metafun*. At least, however, transparency (actually opacity), shading (gradient colors) and transparency group are fully supported, and `outlinetext` is supported by our own alternative `mpliboutlinetext` (see below § 1.2.9). You can try other effects as well, though we did not fully 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 pallellel with this functionality, we support optional argument of instance name for `\everymplib` and `\everyendmplib`, affecting only those `mplibcode` environments of the same name. Unnamed `\everymplib` affects not only those instances with no name, but also those with name but with no corresponding `\everymplib`. The syntax is:

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

1.1.11 Verbatim input

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

1.1.12 $\TeX$ dimen

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

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



1.1.13 $\TeX$ color

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

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

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

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



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

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

1.1.14 `\mpfig ... \endmpfig`

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

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

and the starred version is roughly the same as follows:

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

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

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



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

1.1.15 About cache files

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

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

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

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

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

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

1.1.16 About figure box metric

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

1.1.17 luamplib.cfg

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

1.1.18 Tagged PDF

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

alt=⟨text⟩ starts a Figure tag by default and sets an alternate text of the figure from the `⟨text⟩`. BBox info will be added automatically to the PDF. This key is needed for ordinary `METAPOST` figures, for which, if no `alt` text is given, a default text will be used with a warning

issued. You can change the alternate text within METAPOST code as well: `VerbatimTeX "\mplibaltttext{<text>}";`

actualtext=`<text>` starts a Span tag implicitly and sets a replacement text (a.k.a. actual text) from the `<text>`. If in vertical mode, horizontal mode will be forced by `\noindent` command.⁵ BBox info will not be added. This key is intended for figures which can be represented by a character or a small sequence of characters. You can change the actual text within METAPOST code as well: `VerbatimTeX "\mplibactualtext{<text>}";`

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

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

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

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

```
\begin{mplibcode}[text]
  beginfig(1)
    draw btex [taggingoff] "$\sqrt{2}$ etex ;
    draw texttext "$\sqrt{3}$" shifted 12down ;
    draw TEX "[taggingoff] "$\sqrt{5}$" shifted 24down ;
    draw maketext "$\sqrt{7}$" shifted 36down ;
    draw mplibgraphicstext "$\sqrt{x}$" shifted 48down ;
  endfig;
\end{mplibcode}
```

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

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

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

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

⁵It is not recommended to personally redefine `\prependtomplibbox`. Apart from using `\mplibforcehmode` or `\mplibnoforcehmode`, the redefinition might be incompatible with `actualtext` key. See § 1.1.1 on these commands.

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

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

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

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

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

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

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

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

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

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

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

1.2 METAPOST

1.2.1 T_EX dimen and color

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

1.2.2 More on T_EX color

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

the `withcolor` command.⁹ For instance:

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

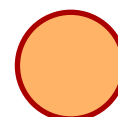
But the result may vary in its color model (gray/rgb/cmyk) according to the given $\TeX$ color. Therefore the example shown above would raise a `METAPOST error: cmykcolor col; should have been declared`. By contrast, `mplibrbgtexcolor` $\langle string \rangle$ always returns rgb-model expressions.

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

1.2.3 Fill and stroke colors

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

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

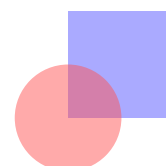


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 Transparency

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

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

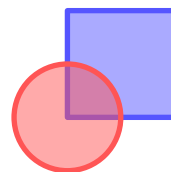


⁹Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color.

1.2.5 Fill and stroke transparency

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

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



1.2.6 Text outline as a text image

`mplibgraphicstext <string>` is a METAPOST operator, the effect of which is similar to that of ConTeXt'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 TeX code, `mplibgraphicstext` will produce better results than ConTeXt 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 $\mathcal{E}\mathcal{T}\mathcal{E}\mathcal{X}$ format.

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

An example:

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



1.2.9 Text outline as a path image

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

A simple example:

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



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

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

1.2.10 Unicode-aware length

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

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

1.2.11 Unicode-aware substring

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

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

1.2.12 Shading

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

- *Textual pictures* as well as paths can have shading effect. The term *textual picture* here means a picture generated by `btex ... etex`, `texttext`, `TEX`, `maketext`, `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.

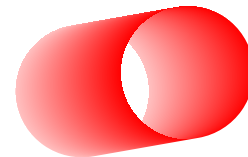
withshadingextend ($\langle boolean \rangle$, $\langle boolean \rangle$) Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

```
\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).

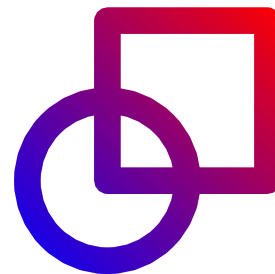
withshadingstroke $\langle \text{string} \rangle$ When the shading object is a $\langle \text{path} \rangle$, the string shall be "yes" or "no": if "yes", we get the path stroked and *then* shaded. It could be more efficient than issuing two sentences.

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

```

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

```



1.2.13 Fading

METAPOST command **withfademethod** makes the color of an object gradiently transparent, a.k.a. *fading*. The syntax is $\langle \text{path} \rangle$ | $\langle \text{picture} \rangle$ **withfademethod** $\langle \text{string} \rangle$, the latter being either "linear" or "circular". Though it is similar to the **withshademethod** from *metafun*, the differences are: (1) the object of fading can be a picture as well as a path; (2) you cannot make gradient colors, but can only make gradient opacity. Technically speaking, this command generates and applies a special kind of masking transparency group described below at § 1.2.17.

Related macros to control optional values are:

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

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

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

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

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

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

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

An example:

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



1.2.14 Tiling pattern

$\text{\TeX}$ macros `\mppattern{<name>} ... \endmppattern` define a tiling pattern cell associated with the *<name>*. METAPOST command `withmppattern`, the syntax being *<cyclic path>* | *<textual picture>* `withmppattern <string>`, will fill the given path or text with the tiling pattern cell of the *<name>* by replicating it horizontally and vertically.¹⁵ As said before at § 1.2.12, the *textual picture* here means basically any text typeset by $\text{\TeX}$, mostly the result of the `btex` command (and its derivatives) or the `infont` operator.

An example:

```
\mppattern{mypatt} % or \begin{mppattern}{mypatt}
```

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, *<cyclic path>* withmppattern *<string>* works as intended only with fill or filldraw command.

Table 1: options for \mppattern

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

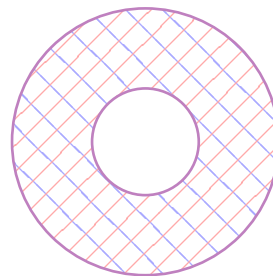
* in string type, numbers are separated by spaces

```

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

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

```



The available options, actually elements of a Lua *table*, are listed in Table 1. For the sake of convenience, the width and height values of the tiling pattern cell will be written down into the log file (depth is always zero). Users can refer to them for option setting.

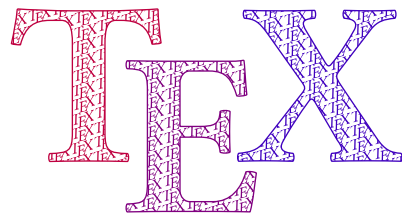
As for *matrix* option, METAPOST code such as "rotated 30 slanted .2" is allowed as well as the string or table of four numbers. You can also set *xshift* and *yshift* values by using 'shifted' operator. But when *xshift* or *yshift* option is explicitly given, they have precedence over the effect of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

When you use special effect such as transparency in a pattern cell, *resources* option is needed: for instance, *resources*="/ExtGState <<MyObj 5 0 R>>". However, as *luamplib* automatically includes the resources of the current page, this option is not needed in most cases.

Option `colored=false` (or `coloured=false`) will generate an uncolored pattern cell which shall have no color at all (i.e. `withoutcolor` command is needed for METAPOST code).¹⁶ Uncolored pattern will be painted later by the color of a METAPOST object. An example:

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

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



A much simpler and efficient way to obtain a similar result (but without colorful characters in this example) is to give a *textual picture* as the operand of `withmppattern`:

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



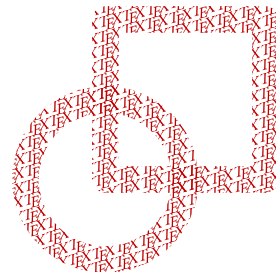
After v2.42.6, we provide an optional macro for both colored and uncolored patterns:

withpatternstroke *<string>* where the string shall be "filldraw", "both", "draw", "yes", or "no":
 "filldraw" or "both" will tile both the fill part and the stroke part; "draw" or "yes" will

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

tile the stroke part only; all other cases will tile the fill part only, which is the default behavior. An example:

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



1.2.15 FormXObject from METAPOST side

As said [before](#) at § 1.1.3, transparency group is available with *plain* as well as *metafun*. It is called *Transparency Group* because the objects contained in the group are composited to produce a single object, so that outer transparency effect, if any, will be applied to the group as a whole, not to the individual objects cumulatively.

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

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

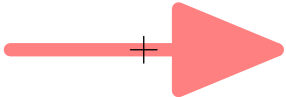
withgroupname $\langle string \rangle$ associates an XObject with the given name. When this command is not appended to the sentence with *asgroup* operator, the default name 'lastmplibgroup' will be used.

`\usemplibgroup{⟨name⟩}` 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 ⟨string⟩` 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 (⟨pair⟩, ⟨pair⟩)` sets the bounding box of the XObject, default value being (llcorner p, urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence ‘withgroupbbox (bot lft llcorner p, top rt urcorner p)’, supposing that the pen was selected by the pickup command.

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

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

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

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

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

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



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



After some experiment, however, it turned out that the best way to get picture shading with transparency group is to give shading effect within an ordinary form XObject and then wrap it up in a transparency group XObject. This sort of double wrapping will be done automatically when you specify ‘wrapped’ as an optional argument of asgroup. Most PDF renderers do render it properly:

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



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

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



```

\mpfig
  draw pic2 ;
\endmpfig
\endmplibgroup

```

```

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

```



1.2.16 Form XObject from T_EX side

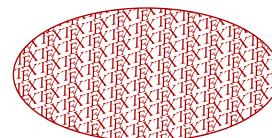
T_EX macros `\mplibgroup{<name>}` ... `\endmplibgroup` are described here in this subsection, as they are deeply related to the `asgroup` operator described just above at § 1.2.15. With these, users can define a transparency group or an ordinary *form XObject* from T_EX side. The syntax is similar to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

An example:¹⁷

```

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

```



```

\usemplibgroup{texpatt}

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

```



Available options are listed in Table 2. Again, the width/height/depth values of the `mplibgroup` will be written down into the log file.

When `asgroup` option is not given or is given as "off", an ordinary *form XObject* will be generated rather than a transparency group. Thus the individual objects, not the XObject as a whole, will be affected by outer transparency command, just like the first figure in the [example](#) above at § 1.2.15.

¹⁷Note that, here again by the option `asgroup="wrapped"`, within a transparency group XObject a tiling pattern is wrapped up in an inner, ordinary XObject. This annoyance is especially for Mac OS Preview which misapplies the transformation matrix to tiling or shading patterns directly wrapped in a transparency group. Most other PDF renderers seem to behave properly even with single wrapping.

Table 2: options for \mplibgroup

Key	Value Type	Explanation
asgroup	<i>string</i>	"", or any comma-separated combination of isolated, knockout, wrapped and off, or "masking"
colorspace	<i>string</i>	/CS entry to a transparency group, such as "/DeviceGray", "/DeviceRGB" or "/DeviceCMYK"
bbox	<i>table</i> or <i>string</i>	llx, lly, urx, ury values*
matrix	<i>table</i> or <i>string</i>	xx, yx, xy, yy values* or MP transform code
resources	<i>string</i>	PDF resources if needed

* in string type, numbers are separated by spaces

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

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

With colorspace option, which is no-op for ordinary formXObject, users can specify the color space of a transparency group. For instance, the mplibgroup will be painted in grayscale model when colorspace="/DeviceGray" is given to an *isolated* transparency group.¹⁸

As shown, you can reuse the mplibgroup using the T_EX command \usemplibgroup or the METAPOST command usemplibgroup. The behavior of these commands is the same as that described [above](#) at § 1.2.15, excepting that the mplibgroup made by T_EX code (not by METAPOST code) respects original height and depth.

1.2.17 Masking

Using the command withmaskinggroup, the mplibgroup (see above § 1.2.16) generated by the option asgroup="masking" (see Table 2) can be utilized as a masking transparency group upon a picture or a path object. The syntax is $\langle picture \rangle$ | $\langle path \rangle$ withmaskinggroup $\langle string \rangle$, the latter being the name of a pre-defined masking group.

Basically, the masking group should be prepared in *grayscale* color model: the area painted with 1 ($\approx$ white: full luminosity) will preserve the full color of the object; the area painted with 0 ($\approx$ black: zero luminosity) will force full transparency, masking it invisibly.¹⁹

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

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

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

¹⁹In fact, colors in other color models are also allowed (such as white, black, red, green, blue). But they will be converted to grayscale model by the PDF renderer, so that "/DeviceGray" is the default value of colorspace option to a masking transparency group (see Table 2 at § 1.2.16).

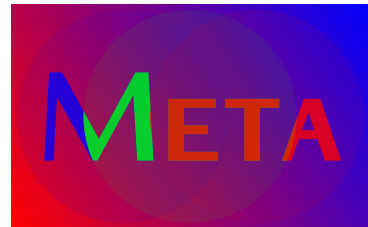
²⁰Color expressions in rgb or cmyk model are also allowed, in accordance with the option colorspace="/DeviceRGB"

An example:

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

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

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



1.2.18 Free-form Gouraud-shaded triangle mesh shading

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

Optional macros for triangle mesh shading:

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

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

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

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

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

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

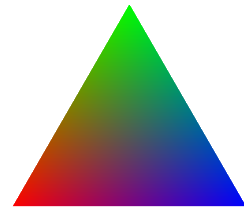
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates).

The third argument is the color for the new vertex.

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

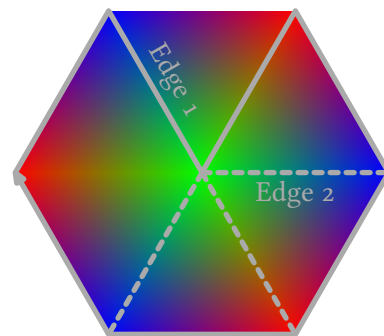
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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



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

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



1.2.19 Lattice-form Gouraud-shaded triangle mesh shading

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

Optional macros for lattice-form shading:

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

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

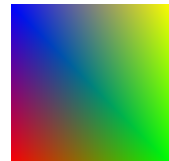
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

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

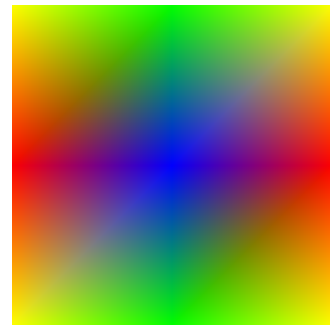
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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

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



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



1.2.20 Coons patch mesh shading

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

Optional macros for Coons patch mesh shading:

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

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

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

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

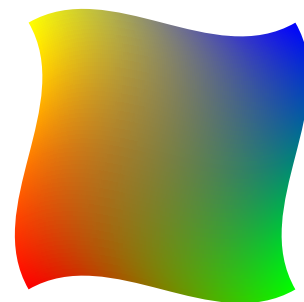
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

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

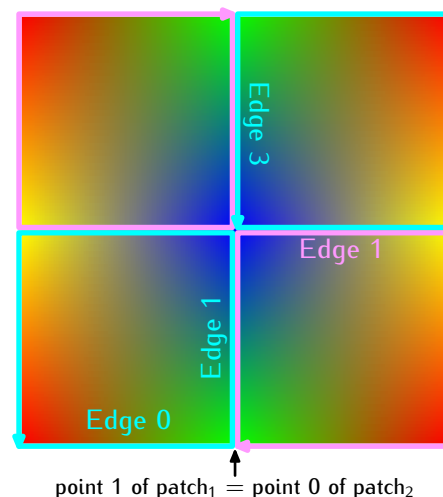
Remaining arguments are two color expressions for the new vertices.

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

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



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



```

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

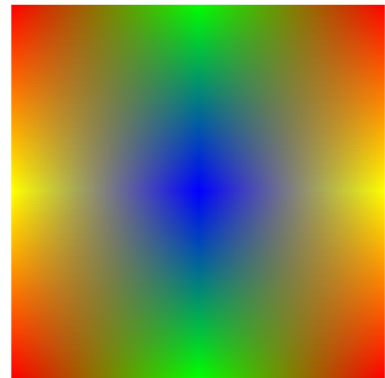
```

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

```

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

```



1.2.21 Tensor-product patch mesh shading

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

Optional macros for tensor-product patch mesh shading:

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

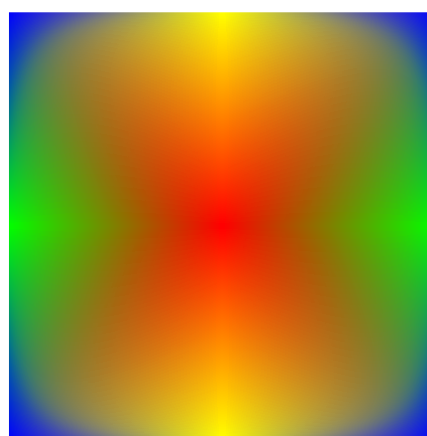
The only difference to withcoonspatchinit macro of Coons patch shading is the second argument inserted for specifying four inner control points. It shall be a path that has four points, or a string composed of eight numerics separated by spaces (x and y coordinates of each point). If the first argument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

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

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

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

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



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

1.3 Lua

1.3.1 runscript ...

A good many METAPOST macros described in this documentation have been implemented using the primitive `runscript`. With `runscript` *⟨string⟩*, you can run a Lua code chunk from METAPOST side and get some METAPOST code returned by Lua if you want. As the functionality is provided by the `mplib` library itself, `luamplib` does not have much to say about it.

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

1.3.2 Accessing METAPOST variables

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

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

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

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

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

3.0

1.3.3 Running a METAPOST code

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

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

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

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

1.3.4 Registering a tiling pattern

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

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

Table 3: elements in luamplib table (partial)

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

The first argument is the register of a box containing a pattern cell, which should be prepared in advance by the user. For instance, `\setbox0=\hbox{\tiny\TeX}`, or corresponding Lua code using `tex.setbox` function; then the argument should be 0.²¹

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

1.3.5 Registering a formXObject

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

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

The first argument is the register of a box prepared in advance by the user. When the contents of the box have been generated from T_EX (not METAPOST) code, please make sure that both of the T_EX macros ‘`MPllx`’ and ‘`MPlly`’ are defined as ‘`0`’ before invoking the Lua function.²²

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

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

2 Implementation

2.1 Lua module

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

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

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version    = "2.42.7",
5   date      = "2026/08/12",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8

```

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

```

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

```

Use our own function for warn/info/err.

```

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

```

```

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

```

Provide a few “shortcuts” expected by the code.

```

50 local tableconcat = table.concat
51 local tableinsert = table.insert
52 local tableunpack = table.unpack
53 local texsprint   = tex.sprint
54 local texgettoks  = tex.gettoks
55 local texgetbox    = tex.getbox
56 local texruntoks   = tex.runtoks
57 if not texruntoks then
58   err("Your LuaTeX version is too old. Please upgrade it to the latest")
59 end
60 local is_defined = token.is_defined
61 local get_macro  = token.get_macro
62 local mplib = require ('mplib')
63 local kpse = require ('kpse')
64 local lfs = require ('lfs')
65 local lfsattributes = lfs.attributes
66 local lfsisdir      = lfs.isdir
67 local lfsmkdir      = lfs.mkdir
68 local lfstouch      = lfs.touch
69 local ioopen        = io.open
70

```

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

```

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

```

92

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

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

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

129 local noneedtoreplace = {
130   ["boxes.mp"] = true, -- ["format.mp"] = true,
131   ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,
132   ["mpost.mp"] = true, ["plain.mp"] = true, ["rboxes.mp"] = true,
133   ["sarith.mp"] = true, ["string.mp"] = true, -- ["TEX.mp"] = true,
134   ["metafun.mp"] = true, ["metafun.mpiv"] = true, ["mp-abck.mpiv"] = true,
135   ["mp-apos.mpiv"] = true, ["mp-asnc.mpiv"] = true, ["mp-bare.mpiv"] = true,

```

```

136 ["mp-base.mpiv"] = true, ["mp-blob.mpiv"] = true, ["mp-butt.mpiv"] = true,
137 ["mp-char.mpiv"] = true, ["mp-chem.mpiv"] = true, ["mp-core.mpiv"] = true,
138 ["mp-crop.mpiv"] = true, ["mp-figs.mpiv"] = true, ["mp-form.mpiv"] = true,
139 ["mp-func.mpiv"] = true, ["mp-grap.mpiv"] = true, ["mp-grid.mpiv"] = true,
140 ["mp-grph.mpiv"] = true, ["mp-idea.mpiv"] = true, ["mp-luas.mpiv"] = true,
141 ["mp-mlib.mpiv"] = true, ["mp-node.mpiv"] = true, ["mp-page.mpiv"] = true,
142 ["mp-shap.mpiv"] = true, ["mp-step.mpiv"] = true, ["mp-text.mpiv"] = true,
143 ["mp-tool.mpiv"] = true, ["mp-cont.mpiv"] = true,
144 }
145 luamplib.noneedtoreplace = noneedtoreplace
146

```

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

```

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

```

Function luamplib.finder

```

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

```

format.mp is much complicated, so specially treated.

```

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

```

```

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

```

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

```

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

```

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

```

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

```

```

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

```

For the main function: process

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

```

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

```

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

```

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

```

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

```

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

```



```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

357

dvipdfmx is supported, though nobody seems to use it.

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

359

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

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

```
361 local catat11 = luatexbase.registernumber("catcodetable@atletter")
```

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

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

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

```
364 end
```

For conversion of sp to bp.

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

366

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

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

```
368 local process_tex_text
```

```
369 do
```

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

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

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

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

```
374     if str then
```

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

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

```
378       local tex_box_id
```

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

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

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

```
382       else
```

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

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

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

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

```
387       end
```

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

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

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

```
392       else
```

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

```

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

```

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

```

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

```

A common function for color functions

```

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

```

```

439 local mplibcolorfmt = {
440   xcolor = [[{\setbox0\hbox{{\color\s\global\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         if not is_defined"extractcolorspecs" then err"needs xcolor package loaded" end
470         run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
471         t = format("%s %s", get_macro"mplibtmpa", get_macro"mplibtmpb":gsub(","," "))
472       end
473       local name, value = t:match("(%a+) (.+)"
474       if name and value then
475         local op = name == "rgb" and "rg" or name == "cmyk" and "k" or name == "gray" and "g"
476         if not op then err"unknown color model" end
477         t = ("%s %s %s %s"):format(value, op, value, op:upper())
478       elseif t:find" cs " then -- spot color raw export
479         name = t:match("^/(.-) cs ")
480         texsprint(ccexplat, {
481           "\\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
482             name, "}{\\pdf_object_ref:n{", name, "}}"
483         })
484       end
485     elseif is_defined"ver@colorspace.sty" and t:find"^/&" then
486       local a,b,c,d = t:match"^(- cs) (- CS) (- scn?) (- SCN?)$"
487       if a and b and c and d then

```

```

488     t = tableconcat({ a, c, b, d }, " ")
489     end
490     end
491     return format('1 withprescript "mpliboverridecolor=%s"', t)
492     end
493     return ""
494 end
495 function process_mplibcolor(str)
496     local res = process_color(str)
497     if res:find" cs " then return res end
498     res = colorsplit(res:match'"mpliboverridecolor=(.)"')
499     return format("(%s)", tableconcat(res, ", "))
500 end
501 end
502
    for \mpdim or mplibdimen
503 local function process_dimen (str)
504     if str then
505         str = str:gsub("{(.+)}", "%1")
506         run_tex_code(format([[ \mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
507         return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
508     end
509     return ""
510 end
511

```

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

```

512 local function process_verbatimtex_text (str)
513     if str then
514         run_tex_code(str)
515     end
516     return ""
517 end
518

```

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.

```

519 local tex_code_pre_mplib = {}
520 luamplib.figid = 1
521 luamplib.in_the_fig = false
522 local function process_verbatimtex_prefig (str)
523     if str then
524         tex_code_pre_mplib[luamplib.figid] = str
525     end
526     return ""
527 end
528 local function process_verbatimtex_infig (str)
529     if str then
530         return format('special "postmplibverbtex=%s";', str)

```

```

531 end
532 return ""
533 end
534

```

For *metafun* format. see issue #79.

```

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

```

metafun 2021-03-09 changes crashes luamplib.

```

540 catcodes = catcodes or {}
541 local catcodes = catcodes
542 catcodes.numbers = catcodes.numbers or {}
543 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
544 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
545 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
546 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
547 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
548 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
549 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
550

```

Now luamplib.runscript

```

551 do
552   local runscript_funcs = {
553     luamplibtext    = process_tex_text,
554     luamplibcolor   = process_mplibcolor,
555     luamplibdimen   = process_dimen,
556     luamplibprefig  = process_verbatimt看_prefig,
557     luamplibinfig   = process_verbatimt看_infig,
558     luamplibverbtex = process_verbatimt看_text,
559   }

```

A function from ConTeXt general.

```

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

```

```

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

```

luamplib.maketext

```

605 luamplib.legacyverbatimimtex = true
606 do

```

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

```

607 local function protecttexcontents (str)
608     return str:gsub("\\%", "\\0PerCent\0")
609         :gsub("%%.-\n", "")
610         :gsub("%%.-$", "")
611         :gsub("%zPerCent%z", "\\0PerCent\0")
612         :gsub("\r.-$", "")
613         :gsub("%s+", " ")
614 end
615 function luamplib.maketext (str, what)
616     if str and str ~= "" then
617         str = protecttexcontents(str)
618         if what == 1 then
619             if not str:find("\\documentclass"..name_e) and
620                 not str:find("\\begin%s*{document}") and

```



```

621         not str:find("\\documentstyle"..name_e) and
622         not str:find("\\usepackage"..name_e) then
623         if luamplib.legacyverbatim then
624             if luamplib.in_the_fig then
625                 return process_verbatim_infig(str)
626             else
627                 return process_verbatim_prefig(str)
628             end
629         else
630             return process_verbatim_text(str)
631         end
632     end
633 else
634     return process_tex_text(str, true) -- bool is for 'char13'
635 end
636 end
637 return ""
638 end
639 end
640

```

luamplib's METAPOST color operators

```

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

```

```

669     "\\color_export:nnN{",
670     str,
671     "}{",
672     rgb and "space-sep-rgb" or "space-sep-cmyk",
673     "\\mplib@tempa",
674     },ccexplat)
675     return get_macro"mplib@tempa":explode()
676   else
677     local name, value = res:match"^(.-) cs (.-) sc"
678     local t = get_macro("spc@ascmyk@"..get_spc_name_obj(name)):explode", " -- mixed not work
679     for i = 1, #t do
680       t[i] = t[i] * value
681     end
682     return rgb and cmyk2rgb(t) or t
683   end
684 end
685 local t = colorsplit(res)
686 if #t == 3 or not rgb then return t end
687 if #t == 4 then return cmyk2rgb(t) end
688 return { t[1], t[1], t[1] }
689 end
690 end
691 luamplib.shadecolor = function (str)
692   local res = process_color(str):match'"mpliboverridecolor=(.+)"'
693   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}

694   local name, value, obj
695   if not is_xcolor(str) then
696     run_tex_code({ "\color_export:nnN{", str, "}{backend}\mplib@tempa" }, ccexplat)
697     name, value = get_macro'mplib@tempa':match'{{(.-)}}{{(.-))}'

```

```

698     local t = res:explode()
699     local refname = t[1]:sub(2,-1)
700     if pdfmode then
701         obj = format("%s 0 R", ltx.pdf.object_id(refname))
702     else
703         run_tex_code({ "\\mplibtmp toks\\expanded{{\\pdf_object_ref:n{", refname, "}}}" }, ccexplat)
704         obj = texgettoks"mplibtmp toks"
705     end
706     else -- colorspace: mixing is allowed only in preamble
707         name, value = res:match"^(.-) cs (.-) sc"
708         name, obj = get_spc_name_obj(name)
709     end
710     return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
711 end
712 return format('(%s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),","), str)
713 end
714

```

luamplib.fillandstrokecolor

```

715 do
716     local function graphictextcolor (col, filldraw)
717         if col:find"^[%d%.:]+$" then
718             col = col:explode"."
719             for i=1,#col do
720                 col[i] = format("%.3f", col[i])
721             end
722             local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
723             col[#col+1] = filldraw == "fill" and op or op:upper()
724             return tableconcat(col," ")
725         end
726         col = process_color(col):match'"mpliboverridecolor=(.+)"'
727         local t = col:explode()
728         local b = filldraw == "fill" and 1 or #t/2+1
729         local e = b == 1 and #t/2 or #t
730         return tableconcat(t," ", b, e)
731     end
732     function luamplib.fillandstrokecolor (fill, stroke)
733         fill = graphictextcolor(fill, "fill")
734         stroke = graphictextcolor(stroke, "stroke")
735         return format('withprescript "mpliboverridecolor=%s %s"', fill, stroke)
736     end
737 end
738

```

Remove trailing zeros for smaller PDF

```

739 local decimals = "%.d+"
740 local function rmzeros(str) return str:gsub("%.?0+$","") end
741

```

common function for mplibgraphictext and mpliboutlinetext

```

742 local function getrulemetric (box, curr, bp)
743   local running = -1073741824
744   local wd,ht,dp = curr.width, curr.height, curr.depth
745   wd = wd == running and box.width or wd
746   ht = ht == running and box.height or ht
747   dp = dp == running and box.depth or dp
748   if bp then
749     return wd/factor, ht/factor, dp/factor
750   end
751   return wd, ht, dp
752 end
753

```

luamplib's mplibgraphicstext operator

```

754 do
755   if not math.round then
756     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
757   end
758   local emboldenfonts = { }
759   local function roundupwidth (f, fb)
760     local wd = math.round(f.size * fb / factor * 10)
761     if wd == 0 and fb ~= 0 then
762       wd = 1
763     end
764     emboldenfonts.width = wd
765     return wd
766   end
767   local function getemboldenwidth (curr, fakebold)
768     local width = emboldenfonts.width
769     if not width then
770       local f
771       local function getglyph(n)
772         while n do
773           if n.head then
774             getglyph(n.head)
775           elseif n.font and n.font > 0 then
776             f = n.font; break
777           end
778           n = node.getnext(n)
779         end
780       end
781       getglyph(curr)
782       width = roundupwidth(font.getcopy(f or font.current()), fakebold)
783     end
784     return width
785   end
786   local function getrulewhatsit (line, wd, ht, dp)
787     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
788     line = line == 0 and "" or ("%f w"):format(line)

```

```

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

```

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

```

804   local tag_update_attrs
805   if is_defined"ver@tagpdf.sty" then
806     tag_update_attrs = function (n, curr)
807       while n do
808         n.attr = curr.attr
809         if n.head then
810           tag_update_attrs(n.head, curr)
811         end
812         n = node.getnext(n)
813       end
814     end
815   else
816     tag_update_attrs = function() end
817   end
818   local function embolden (box, curr, fakebold)
819     local head = curr
820     while curr do
821       if curr.head then
822         curr.head = embolden(curr, curr.head, fakebold)
823       elseif curr.replace then
824         curr.replace = embolden(box, curr.replace, fakebold)
825       elseif curr.leader then
826         if curr.leader.head then
827           curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
828         elseif curr.leader.id == node.id"rule" then
829           local glue = node.effective_glue(curr, box)
830           local line = getemboldenwidth(curr, fakebold)
831           local wd,ht,dp = getrulemetric(box, curr.leader)
832           if box.id == node.id"hlist" then
833             wd = glue
834           else
835             ht, dp = 0, glue

```

```

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

```

```

885     end
886     return head
887 end
888 luamplib.graphicstext = function (text, fakebold, fc, dc)
889     local fmt = process_tex_text(text):sub(1,-2)
890     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
891     emboldenfonts.width = nil
892     local box = texgetbox(id)
893     box.head = embolden(box, box.head, fakebold)
894     local colors = luamplib.fillandstrokecolor(fc, dc)
895     return format('%s %s)', fmt, colors)
896 end
897 end
898

```

luamplib's mplibglyph operator

```

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

```



```

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

```

```

981     local fid = tonumber(f) or font.id(f)
982     if fid > 0 then
983         local fontdata = font.getfont(fid) or font.getcopy(fid)
984         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
985         instance = fontdata.specification and fontdata.specification.instance
986             or fontdata.shared and fontdata.shared.features.axis
987         filename = filename and filename:gsub("^harfloaded:", "")
988     else
989         local name
990         f = f:match("^s*(.+)s*$")
991         name, subfont, instance = f:match"(.)%((%d+)%)%[(.-)]%"
992         if not name then
993             name, instance = f:match"(.)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
994         end
995         if not name then
996             name, subfont = f:match"(.)%((%d+)%)$" -- Times.ttc(2)
997         end
998         name = name or f
999         subfont = (subfont or 0)+1
1000        instance = instance and instance:lower()
1001        for _, ftype in ipairs{"opentype", "truetype"} do
1002            filename = kpse.find_file(name, ftype.." fonts")
1003            if filename then
1004                kind = ftype; break
1005            end
1006        end
1007    end
1008    if kind ~= "opentype" and kind ~= "truetype" then
1009        f = fid and fid > 0 and tex.fontname(fid) or f
1010        if kpse.find_file(f, "tfm") then
1011            return format("glyph %s of %q", tonumber(c) or format("%q", c), f)
1012        else
1013            filename = kpse.find_file(f, "type1 fonts")
1014            if filename then
1015                kind = "type1" -- there's bug in processing cmr family
1016            else
1017                return mperr"font not found"
1018            end
1019        end
1020    end
1021    local time = lfsattributes(filename, "modification")

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

1022    local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1023    local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
1024    local newname = format("%s/%s.lua", cachedir or outputdir(), h)
1025    local newtime = lfsattributes(newname, "modification") or 0

```

```

1026     if time == newtime then
1027         shapedata = require(newname)
1028     end
1029     if not shapedata then
1030         if fonts then
1031             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1032             shapedata = handler.readers.loadshapes(filename, subfont, instance)
1033         end
1034         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1035         table.tofile(newname, shapedata, "return")
1036         lfstouch(newname, time, time)
1037     end
1038     local gid = tonumber(c)
1039     if not gid then
1040         local uni = utf8.codepoint(c)
1041         for i,v in pairs(shapedata.glyphs) do
1042             if c == v.name or uni == v.unicode then
1043                 gid = i; break
1044             end
1045         end
1046     end
1047     if not gid then return mperr"cannot get GID (glyph id)" end
1048     local fac = 1000 / (shapedata.units or 1000)
1049     local t = shapedata.glyphs[gid]; t = t and t.segments
1050     if not t then return "image()" end
1051     for i,v in ipairs(t) do
1052         if type(v) == "table" then
1053             for ii,vv in ipairs(v) do
1054                 if type(vv) == "number" then
1055                     t[i][ii] = format("%.0f", vv * fac)
1056                 end
1057             end
1058         end
1059     end
1060     local result, towarn = glyphimage(t, shapedata.format or kind)
1061     if towarn then
1062         warn("mplibglyph %s not working properly. Use glyph instead", f)
1063     end
1064     return result
1065 end
1066 end
1067

```

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

```

1068 do
1069     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1070         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1071     local outline_horz, outline_vert
1072     function outline_vert (res, box, curr, xshift, yshift)

```

```

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

```

```

1122         iy = b2u and iy or -iy
1123         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1124         for i=1,n do
1125             res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1126             dy = dy + hd + iy
1127         end
1128     end
1129 end
1130 end
1131 dy = dy + (b2u and vwidth or -vwidth)
1132 elseif curr.id == node.id"kern" then
1133     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1134 elseif curr.id == node.id"vlist" then
1135     dy = dy + (b2u and curr.depth or -curr.height)/factor
1136     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1137     dy = dy + (b2u and curr.height or -curr.depth)/factor
1138 elseif curr.id == node.id"hlist" then
1139     dy = dy + (b2u and curr.depth or -curr.height)/factor
1140     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1141     dy = dy + (b2u and curr.height or -curr.depth)/factor
1142 end
1143 curr = node.getnext(curr)
1144 end
1145 return res
1146 end
1147 function outline_horz (res, box, curr, xshift, yshift, discwd)
1148     local r2l = box.dir == "TRT"
1149     local dx = r2l and (discwd or box.width/factor) or 0
1150     local dirs = { { dir = r2l, dx = dx } }
1151     while curr do
1152         if curr.id == node.id"dir" then
1153             local sign, dir = curr.dir:match"(.)(...)"
1154             local level, newdir = curr.level, r2l
1155             if sign == "+" then
1156                 newdir = dir == "TRT"
1157                 if r2l ~= newdir then
1158                     local n = node.getnext(curr)
1159                     while n do
1160                         if n.id == node.id"dir" and n.level+1 == level then break end
1161                         n = node.getnext(n)
1162                     end
1163                     n = n or node.tail(curr)
1164                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1165                 end
1166                 dirs[level] = { dir = r2l, dx = dx }
1167             else
1168                 local level = level + 1
1169                 newdir = dirs[level].dir
1170                 if r2l ~= newdir then

```

```

1171         dx = dirs[level].dx
1172     end
1173 end
1174 r2l = newdir
1175 elseif curr.char and curr.font and curr.font > 0 then
1176     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1177     local gid = ft.characters[curr.char].index or curr.char
1178     local scale = ft.size / factor / 1000
1179     local slant = (ft.slant or 0)/1000
1180     local extend = (ft.extend or 1000)/1000
1181     local squeeze = (ft.squeeze or 1000)/1000
1182     local expand = 1 + (curr.expansion_factor or 0)/1000000
1183     local xscale, yscale = scale * extend * expand, scale * squeeze
1184     dx = dx - (r2l and curr.width/factor*expand or 0)
1185     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1186     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1187     local vertical = ""
1188     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1189         if ft.shared.features.vertical then -- luatexko
1190             vertical = "rotated 90"
1191             local data = ft.characters[curr.char] or { }
1192             if ft.hb then
1193                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1194                 local charraise = (ft.luatexko_charraise or 0)/factor
1195                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1196             else
1197                 local cmds = data.commands or { {0,0}, {0,0} }
1198                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1199                 xpos, ypos = xpos + hoff, ypos + voff
1200             end
1201         elseif curr ~= box.head then -- luatexja
1202             vertical = "rotated 90"
1203             local en = ft.parameters.quad/factor/2
1204             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1205         end
1206     end
1207     local image
1208     if ft.format == "opentype" or ft.format == "truetype" then
1209         image = luamplib.glyph(curr.font, gid)
1210     else
1211         local name, scale = ft.name, 1
1212         local vf = font.read_vf(name, ft.size)
1213         if vf and vf.characters[gid] then
1214             local cmds = vf.characters[gid].commands or {}
1215             for _,v in ipairs(cmds) do
1216                 if v[1] == "char" then
1217                     gid = v[2]
1218                 elseif v[1] == "font" and vf.fonts[v[2]] then
1219                     name = vf.fonts[v[2]].name

```

```

1220         scale = vf.fonts[v[2]].size / ft.size
1221     end
1222 end
1223 end
1224     image = format("glyph %s of %q scaled %f", gid, name, scale)
1225 end
1226     res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1227         #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1228     dx = dx + (r2l and 0 or curr.width/factor*expand)
1229 elseif curr.replace then
1230     local width = node.dimensions(curr.replace)/factor
1231     dx = dx - (r2l and width or 0)
1232     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1233     dx = dx + (r2l and 0 or width)
1234 elseif curr.id == node.id"rule" then
1235     local wd, ht, dp = getrulemetric(box, curr, true)
1236     if wd ~= 0 then
1237         local hd = ht + dp
1238         dx = dx - (r2l and wd or 0)
1239         if hd ~= 0 and curr.subtype == 0 then
1240             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1241         end
1242         dx = dx + (r2l and 0 or wd)
1243     end
1244 elseif curr.id == node.id"glue" then
1245     local width = node.effective_glue(curr, box)/factor
1246     dx = dx - (r2l and width or 0)
1247     if curr.leader then
1248         local curr, kind = curr.leader, curr.subtype
1249         if curr.id == node.id"rule" then
1250             local wd, ht, dp = getrulemetric(box, curr, true)
1251             local hd = ht + dp
1252             if hd ~= 0 then
1253                 wd = width
1254                 if wd ~= 0 and curr.subtype == 0 then
1255                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
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
1267                     n = math.floor((width-diff) / wd)
1268                     dx = dx + (r2l and -diff-wd or diff)

```

```

1269         else
1270             n = math.floor(width / wd)
1271             if kind == 101 then
1272                 local side = width % wd / 2
1273                 dx = dx + (r2l and -side-wd or side)
1274             elseif kind == 102 then
1275                 ix = width % wd / (n+1)
1276                 dx = dx + (r2l and -ix-wd or ix)
1277             end
1278         end
1279         wd = r2l and -wd or wd
1280         ix = r2l and -ix or ix
1281         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1282         for i=1,n do
1283             res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1284             dx = dx + wd + ix
1285         end
1286     end
1287 end
1288 end
1289 dx = dx + (r2l and 0 or width)
1290 elseif curr.id == node.id"kern" then
1291     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1292 elseif curr.id == node.id"math" then
1293     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1294 elseif curr.id == node.id"vlist" then
1295     dx = dx - (r2l and curr.width/factor or 0)
1296     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1297     dx = dx + (r2l and 0 or curr.width/factor)
1298 elseif curr.id == node.id"hlist" then
1299     dx = dx - (r2l and curr.width/factor or 0)
1300     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1301     dx = dx + (r2l and 0 or curr.width/factor)
1302 end
1303 curr = node.getnext(curr)
1304 end
1305 return res
1306 end
1307 function luamplib.outlinetext (text)
1308     local fmt = process_tex_text(text)
1309     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1310     local box = texgetbox(id)
1311     local res = outline_horz({ }, box, box.head, 0, 0)
1312     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1313     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1314 end
1315 end
1316

```


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

```

1317 function luamplib.getunicodegraphemes (s)
1318   local t = { }
1319   local graphemes = require'lua-uni-graphemes'
1320   for _, _, c in graphemes.graphemes(s) do
1321     table.insert(t, c)
1322   end
1323   return t
1324 end
1325 function luamplib.unicodesubstring (s,b,e,grph)
1326   local tt, t, step = { }
1327   if grph then
1328     t = luamplib.getunicodegraphemes(s)
1329   else
1330     t = { }
1331     for _, c in utf8.codes(s) do
1332       table.insert(t, utf8.char(c))
1333     end
1334   end
1335   if b <= e then
1336     b, step = b+1, 1
1337   else
1338     e, step = e+1, -1
1339   end
1340   for i = b, e, step do
1341     table.insert(tt, t[i])
1342   end
1343   s = table.concat(tt):gsub("'", "'&ditto'")
1344   return string.format("%s", s)
1345 end
1346

```

METAPOST preambles

```

1347 luamplib.preambles = {
1348   preamble = [[
1349 boolean mplib ; mplib := true ;
1350 let dump = endinput ;
1351 let normalfontsize = fontsize;
1352 input %s ;
1353 ]],
1354   mplibcode = [[
1355 texscriptmode := 2;
1356 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1357 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1358 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1359 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1360 if known context_mlib:
1361   defaultfont := "cmtt10";
1362   let infont = normalinfont;

```

```

1363 let fontsize = normalfontsize;
1364 vardef thelabel@#(expr p,z) =
1365   if string p :
1366     thelabel@#(p infont defaultfont scaled defaultscale,z)
1367   else :
1368     p shifted (z + labeloffset*mfun_laboff@# -
1369       (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1370       (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1371   fi
1372 enddef;
1373 else:
1374 vardef texttext@# primary t = rawtexttext (t) enddef;
1375 def message expr t =
1376   if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1377 enddef;
1378 def withtransparency (expr a, t) =
1379   withprescript "tr_alternative=" & if numeric a: decimal fi a
1380   withprescript "tr_transparency=" & decimal t
1381 enddef;
1382 vardef ddecimal primary p =
1383   decimal xpart p & " " & decimal ypart p
1384 enddef;
1385 vardef boundingbox primary p =
1386   if (path p) or (picture p) :
1387     llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1388   else :
1389     origin
1390   fi -- cycle
1391 enddef;
1392 fi
1393 def resolvedcolor(expr s) =
1394   runscript("return luamplib.shadecolor('"&s &"')")
1395 enddef;
1396 def colordecimals primary c =
1397   if cmykcolor c:
1398     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1399     decimal yellowpart c & ":" & decimal blackpart c
1400   elseif rgbcolor c:
1401     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1402   elseif string c:
1403     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1404   else:
1405     decimal c
1406   fi
1407 enddef;
1408 def externalfigure primary filename =
1409   draw rawtexttext("\includegraphics{"& filename &"}")
1410 enddef;
1411 def TEX = texttext enddef;

```

```

1412 def mplibtexcolor primary c =
1413   runscript("return luamplib.gettexcolor('& c &'")
1414 enddef;
1415 def mplibrbgtexcolor primary c =
1416   runscript("return luamplib.gettexcolor('& c &', 'rgb')")
1417 enddef;
1418 def mplibgraphictext primary t =
1419   begingroup;
1420   mplibgraphictext_ (t)
1421 enddef;
1422 def mplibgraphictext_ (expr t) text rest =
1423   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1424   fb, fc, dc, graphictextpic, alsoordoublepath;
1425   picture graphictextpic; graphictextpic := nullpicture;
1426   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1427   let scale = scaled;
1428   def fakebold primary c = hide(fb:=c;) enddef;
1429   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1430   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1431   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1432   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1433   addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1434   def fakebold primary c = enddef;
1435   let fillcolor = fakebold; let drawcolor = fakebold;
1436   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1437   image(draw runscript("return luamplib.graphictext([==["&t&"]==], "
1438     & decimal fb & ", "& fc & ", "& dc & "'") rest;))
1439   endgroup;
1440 enddef;
1441 def mplibglyph expr c of f =
1442   runscript (
1443     "return luamplib.glyph('"
1444     & if numeric f: decimal fi f
1445     & " ', '"
1446     & if numeric c: decimal fi c
1447     & " ')"
1448   )
1449 enddef;
1450 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1451 def mplibdrawglyph expr g =
1452   luamplib_tmp_num_ := 0;
1453   for item within g:
1454     fill pathpart item
1455     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1456   endfor
1457 enddef;
1458 let mplibfillglyph = mplibdrawglyph;
1459 def mplibstrokeglyph expr g =
1460   luamplib_tmp_num_ := 0;

```

```

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

```

```

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

```

```

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

```

```

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

```

```

1657     decimal (xpart llcorner p -1/4) & ":" &
1658     decimal (ypart llcorner p -1/4) & ":" &
1659     decimal (xpart urcorner p +1/4) & ":" &
1660     decimal (ypart urcorner p +1/4)
1661     withprescript "gr_state=start"
1662     withprescript "gr_type=" & s;
1663     draw p withprescript "sh_in_xobj=yes";
1664     draw center p withprescript "gr_state=stop";
1665 )
1666 enddef;
1667 def withgroupbbox (expr a,b) =
1668   withprescript "mplibgroupbbox=" &
1669   decimal xpart a & ":" &
1670   decimal ypart a & ":" &
1671   decimal xpart b & ":" &
1672   decimal ypart b
1673 enddef;
1674 def withgroupname expr s =
1675   withprescript "mplibgroupname=" & s
1676 enddef;
1677 def usemplibgroup primary s =
1678   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s & "\endcsname}")
1679   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1680 enddef;
1681 path    mplib_shade_path ;
1682 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1683 numeric mplib_shade_fx, mplib_shade_fy ;
1684 numeric mplib_shade_lx, mplib_shade_ly ;
1685 numeric mplib_shade_nx, mplib_shade_ny ;
1686 numeric mplib_shade_dx, mplib_shade_dy ;
1687 numeric mplib_shade_tx, mplib_shade_ty ;
1688 primarydef p withshadingmethod m =
1689   p
1690   if picture p :
1691     withprescript "sh_operand_type=picture"
1692     if textual p or (length p > 1):
1693       withprescript "sh_transform=no"
1694       mplib_with_shade_method (boundingbox p, m)
1695     else:
1696       withprescript "sh_transform=yes"
1697       mplib_with_shade_method (pathpart p, m)
1698     fi
1699   else :
1700     withprescript "sh_transform=yes"
1701     mplib_with_shade_method (p, m)
1702   fi
1703 enddef;
1704 def mplib_with_shade_method (expr p, m) =
1705   hide(mplib_with_shade_method_analyze(p))

```



```

1706 withprescript "sh_domain=0 1"
1707 withprescript "sh_color=into"
1708 withprescript "sh_color_a=" & colordecimals white
1709 withprescript "sh_color_b=" & colordecimals black
1710 withprescript "sh_first=" & ddecimal point 0 of p
1711 withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1712 withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1713 if m = "linear" :
1714   withprescript "sh_type=linear"
1715   withprescript "sh_factor=1"
1716   withprescript "sh_center_a=" & ddecimal llcorner p
1717   withprescript "sh_center_b=" & ddecimal urcorner p
1718 elseif m = "coons":
1719   withprescript "sh_type=coons"
1720   withprescript "sh_transform=no"
1721 elseif m = "triangle":
1722   withprescript "sh_type=triangle"
1723   withprescript "sh_transform=no"
1724 elseif m = "lattice":
1725   withprescript "sh_type=lattice"
1726   withprescript "sh_transform=no"
1727 elseif m = "tensor":
1728   withprescript "sh_type=tensor"
1729   withprescript "sh_transform=no"
1730   withprescript "sh_tensor_path=" &
1731     ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1732     ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1733     ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1734     ddecimal 1/4[point 3 of p, point 1 of p]
1735 else :
1736   withprescript "sh_type=circular"
1737   withprescript "sh_factor=1.2"
1738   withprescript "sh_center_a=" & ddecimal center p
1739   withprescript "sh_center_b=" & ddecimal center p
1740   withprescript "sh_radius_a=" & decimal 0
1741   withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1742 fi
1743 enddef;
1744 def withlatticeverticesperrow expr n =
1745   withprescript "sh_lattice_perrow=" & decimal n
1746 enddef;
1747 def withlatticeverticesdata (text t) =
1748   hide(luamplib_tmp_num_ := 0;)
1749   withprescript "sh_lattice_data=" &
1750   for item = t:
1751     if odd(incr luamplib_tmp_num_):
1752       if pair item: ddecimal fi item & " " &
1753       fi
1754   endfor ""

```

```

1755 hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1756 for item = t:
1757   if not odd(incr luamplib_tmp_num_):
1758     withshadingstep( withshadingcolors(item,item) )
1759   fi
1760 endfor
1761 enddef;
1762 def withtrianglepatchinit (expr p, a, b, c) =
1763   if string p:
1764     withtrianglepatchnext(0, p , a)
1765     withtrianglepatchnext(0, "", b)
1766     withtrianglepatchnext(0, "", c)
1767   else:
1768     withtrianglepatchnext(0, point 0 of p, a)
1769     withtrianglepatchnext(0, point 1 of p, b)
1770     withtrianglepatchnext(0, point 2 of p, c)
1771   fi
1772 enddef;
1773 def withtrianglepatchnext (expr f, p, a) =
1774   withshadingstep(
1775     withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1776     withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1777       if string p: p else: ddecimal p fi
1778     withshadingcolors (a, a)
1779   )
1780 enddef;
1781 def withcoonspatchinit (expr p, a, b, c, d) =
1782   withshadingstep(
1783     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1784     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1785       if string p: p
1786       else:
1787         ddecimal point      0 of p & " " &
1788         ddecimal postcontrol 0 of p & " " &
1789         ddecimal precontrol  1 of p & " " &
1790         ddecimal point      1 of p & " " &
1791         ddecimal postcontrol 1 of p & " " &
1792         ddecimal precontrol  2 of p & " " &
1793         ddecimal point      2 of p & " " &
1794         ddecimal postcontrol 2 of p & " " &
1795         ddecimal precontrol  3 of p & " " &
1796         ddecimal point      3 of p & " " &
1797         ddecimal postcontrol 3 of p & " " &
1798         ddecimal precontrol  0 of p
1799       fi
1800     withshadingcolors (a, b)
1801   )
1802   withshadingstep ( withshadingcolors (c, d) )
1803 enddef;

```

```

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

```

```

1853     if string p: p & " " & q
1854     else:
1855         ddecimal postcontrol 1 of p & " " &
1856         ddecimal precontrol 2 of p & " " &
1857         ddecimal point 2 of p & " " &
1858         ddecimal postcontrol 2 of p & " " &
1859         ddecimal precontrol 3 of p & " " &
1860         ddecimal point 3 of p & " " &
1861         ddecimal postcontrol 3 of p & " " &
1862         ddecimal precontrol 0 of p & " " &
1863         ddecimal point 0 of q & " " &
1864         ddecimal point 1 of q & " " &
1865         ddecimal point 2 of q & " " &
1866         ddecimal point 3 of q
1867     fi
1868     withshadingcolors (a, b)
1869 )
1870 enddef;
1871 def mplib_with_shade_method_analyze(expr p) =
1872     mplib_shade_path := p ;
1873     mplib_shade_step := 1 ;
1874     mplib_shade_fx := xpart point 0 of p ;
1875     mplib_shade_fy := ypart point 0 of p ;
1876     mplib_shade_lx := mplib_shade_fx ;
1877     mplib_shade_ly := mplib_shade_fy ;
1878     mplib_shade_nx := 0 ;
1879     mplib_shade_ny := 0 ;
1880     mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1881     mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1882     for i=1 upto length(p) :
1883         mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1884         mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1885         if mplib_shade_tx > mplib_shade_dx :
1886             mplib_shade_nx := i + 1 ;
1887             mplib_shade_lx := xpart point i of p ;
1888             mplib_shade_dx := mplib_shade_tx ;
1889         fi ;
1890         if mplib_shade_ty > mplib_shade_dy :
1891             mplib_shade_ny := i + 1 ;
1892             mplib_shade_ly := ypart point i of p ;
1893             mplib_shade_dy := mplib_shade_ty ;
1894         fi ;
1895     endfor ;
1896 enddef;
1897 vardef mplib_max_radius(expr p) =
1898     max (
1899         (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1900         (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1901         (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),

```

```

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

```

```

1951 def withshadingdomain expr d =
1952   withprescript "sh_domain=" & ddecimal d
1953 enddef;
1954 def withshadingfactor expr f =
1955   withprescript "sh_factor=" & decimal f
1956 enddef;
1957 def withshadingfraction expr a =
1958   if mplib_shade_step > 0 :
1959     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1960   fi
1961 enddef;
1962 let withfadefraction = withshadingfraction;
1963 def withshadingcolors (expr a, b) =
1964   if mplib_shade_step > 0 :
1965     withprescript "sh_color=into"
1966     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1967     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1968   else :
1969     withprescript "sh_color=into"
1970     withprescript "sh_color_a=" & colordecimals a
1971     withprescript "sh_color_b=" & colordecimals b
1972   fi
1973 enddef;
1974 let withfadeopacity = withshadingcolors;
1975 def withshadingstroke expr a =
1976   withprescript "sh_stroking=" & a
1977 enddef;
1978 def withshadingmatrix expr s =
1979   withprescript "sh_matrix=" & s
1980 enddef;
1981 let withfadematrix = withshadingmatrix;
1982 def mpliblength primary t =
1983   runscript("return utf8.len[==[" & t & "]==]")
1984 enddef;
1985 def mplibsubstring expr p of t =
1986   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1987     & decimal xpart p & ",",
1988     & decimal ypart p & ")")
1989 enddef;
1990 def mplibuclength primary t =
1991   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
1992 enddef;
1993 def mplibucsubstring expr p of t =
1994   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1995     & decimal xpart p & ",",
1996     & decimal ypart p & ",true)")
1997 enddef;
1998 ]],
1999 legacyverbatimtex = [[

```

```

2000 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
2001 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
2002 let VerbatimTeX = specialVerbatimTeX;
2003 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
2004 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
2005 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
2006 "runscript(" &ditto&
2007 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
2008 "luamplib.in_the_fig=false" &ditto& ");";
2009 ]],
2010 texttextlabel = [[
2011 let luampliboriginalinfont = infont;
2012 primarydef s infont f =
2013   if (s < char 32)
2014     or (s = char 35) % #
2015     or (s = char 36) % $
2016     or (s = char 37) % %
2017     or (s = char 38) % &
2018     or (s = char 92) % \
2019     or (s = char 94) % ^
2020     or (s = char 95) % _
2021     or (s = char 123) % {
2022     or (s = char 125) % }
2023     or (s = char 126) % ~
2024     or (s = char 127) :
2025     s luampliboriginalinfont f
2026   else :
2027     rawtexttext(s)
2028   fi
2029 enddef;
2030 def fontsize expr f =
2031   begingroup
2032   save size; numeric size;
2033   size := mplibdimen("1em");
2034   if size = 0: 10pt else: size fi
2035   endgroup
2036 enddef;
2037 ]],
2038 }
2039
process_mplibcode
When \mplibverbatim is enabled, do not expand mplibcode data.
2040 luamplib.verbatiminput = false
2041 luamplib.everymplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2042 luamplib.everyendmplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2043 function luamplib.process_mplibcode (data, instanceName)
2044   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2045 if luamplib.legacyverbatim then
2046   luamplib.figid, tex_code_pre_mplib = 1, {}
2047 end
2048 local everymplib = luamplib.everymplib[instancename]
2049 local everyendmplib = luamplib.everyendmplib[instancename]
2050 data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
2051 :gsub("\r", "\n")

```

These five lines are needed for mplibverbatim mode.

```

2052 if luamplib.verbatiminput then
2053   data = data:gsub("\mpcolor{s+(-%b{})}", "mplibcolor(\"%1\")")
2054   :gsub("\mpdim{s+(%b{})}", "mplibdimen(\"%1\")")
2055   :gsub("\mpdim{s+(\%a+)", "mplibdimen(\"%1\")")
2056   :gsub(btex_etex, "btex %1 etex ")
2057   :gsub(verbatimtex_etex, "verbatimtex %1 etex;")
2058 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.

```

2059   local t = { } -- to store btex, verbatimtex, string
2060   data = data:gsub(btex_etex, function(str)
2061     t[#t+1] = str
2062     return format("btex \unexpanded{!l!u!a!s!m!p!l!} etex ", #t) -- space
2063   end)
2064   :gsub(verbatimtex_etex, function(str)
2065     t[#t+1] = str
2066     return format("verbatimtex \unexpanded{!l!u!a!s!m!p!l!} etex;", #t) -- semicolon
2067   end)
2068   :gsub('"(.)"', function(str)
2069     t[#t+1] = str
2070     return format("\unexpanded{!l!u!a!s!m!p!l!}"', #t)
2071   end)
2072   :gsub("\%%", "\0PerCent\0")
2073   :gsub("%%.-\n", "\n")
2074   :gsub("%zPerCent%z", "\%")
2075   run_tex_code(format("\mplibtmptoks\expandafter{\expanded{%s}}", data))
2076   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

2077   :gsub("##", "#")
2078   :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2079 end
2080 process(data, instancename)
2081 end
2082

```

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.

```

2083 local figcontents = { post = { } }

```



```

2084 local function put2output(a,...)
2085   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2086 end
2087 local function pdf_startfigure(n,llx,lly,urx,ury)
2088   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2089 end
2090 local function pdf_stopfigure()
2091   put2output("\mplibstoptoPDF")
2092 end

```

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

```

2093 local function pdf_literalcode (...)
2094   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2095 end
2096 local start_pdf_code = pdfmode
2097   and function() pdf_literalcode"q" end
2098   or function() put2output"\special{pdf:bcontent}" end
2099 local stop_pdf_code = pdfmode
2100   and function() pdf_literalcode"Q" end
2101   or function() put2output"\special{pdf:econtent}" end
2102

```

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

```

2103 local function put_tex_boxes (object,prescript)
2104   local box = prescript.mplibtexboxid:explode":"
2105   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2106   if n and tw and th then
2107     local op = object.path
2108     local first, second, fourth = op[1], op[2], op[4]
2109     local tx, ty = first.x_coord, first.y_coord
2110     local sx, rx, ry, sy = 1, 0, 0, 1
2111     if tw ~= 0 then
2112       sx = (second.x_coord - tx)/tw
2113       rx = (second.y_coord - ty)/tw
2114       if sx == 0 then sx = 0.00001 end
2115     end
2116     if th ~= 0 then
2117       sy = (fourth.y_coord - ty)/th
2118       ry = (fourth.x_coord - tx)/th
2119       if sy == 0 then sy = 0.00001 end
2120     end
2121

```

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

```

2122   local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2123   put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2124 end
2125 end
2126

```

Colors

```

2127 local function do_preobj_CR(object,prescript)
2128   if object.postscript == "collect" then return end
2129   local override = prescript and prescript.mpliboverridecolor
2130   if override then
2131     pdf_literalcode(override)
2132   else
2133     local cs = object.color
2134     if cs and #cs > 0 then
2135       pdf_literalcode(luamplib.colorconverter(cs))
2136     end
2137   end
2138 end
2139

```

For transparency, shading, fading, and pattern

```

2140 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2141 local pdfobjs, pdfetcs = {}, {}
2142 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2143 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2144 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
2145 local update_pdfobjs
2146 if pdfmode then
2147   function update_pdfobjs (os, stream)
2148     local key = os
2149     if stream then key = key..stream end
2150     local on = key and pdfobjs[key]
2151     if on then
2152       return on,false
2153     end
2154     if stream then
2155       on = pdf.immediateobj("stream",stream,os)
2156     elseif os then
2157       on = pdf.immediateobj(os)
2158     else
2159       on = pdf.reserveobj()
2160     end
2161     if key then
2162       pdfobjs[key] = on
2163     end
2164     return on,true
2165   end
2166 else
2167   function update_pdfobjs (os, stream, hex)
2168     local key = os
2169     if stream then key = key..stream end
2170     local on = key and pdfobjs[key]
2171     if on then
2172       return on,false
2173     end

```

```

2174     on = pdfetcs.cnt or 1
2175     if stream then
2176         if hex then
2177             texsprintf(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2178         else
2179             texsprintf(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2180         end
2181     elseif os then
2182         texsprintf(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2183     else
2184         texsprintf(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2185     end
2186     pdfetcs.cnt = on + 1
2187     if key then
2188         pdfobjs[key] = on
2189     end
2190     return on,true
2191 end
2192 end
2193 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2194 if pdfmode then
2195     pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2196     local getpagers = pdfetcs.getpagers
2197     local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2198     local initialize_resources = function(name)
2199         local tabname = format("%s_res",name)
2200         pdfetcs[tabname] = { }
2201         if luatexbase.callbacktypes.finish_pdffile then -- ltuatex
2202             local obj = pdf.reserveobj()
2203             setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2204             luatexbase.add_to_callback("finish_pdffile", function()
2205                 pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
2206             end,
2207             format("luamplib.%s.finish_pdffile",name))
2208         end
2209     end
2210     pdfetcs.fallback_update_resources = function (name, res)
2211         local tabname = format("%s_res",name)
2212         if not pdfetcs[tabname] then
2213             initialize_resources(name)
2214         end
2215         if luatexbase.callbacktypes.finish_pdffile then
2216             local t = pdfetcs[tabname]
2217             t[#t+1] = res
2218         else
2219             local tpr, n = getpagers() or "", 0
2220             tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2221             if n == 0 then
2222                 tpr = format("%s/%s<<%s>>", tpr, name, res)

```

```

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

```

Transparency

```

2246 local function add_extgs_resources (on, new)
2247   local key = format("MPLibTr%s", on)
2248   if new then
2249     local val = format(pdfetcs.resfmt, on)
2250     if pdfmanagement then
2251       texsprint {
2252         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2253       }
2254     else
2255       local tr = format("/%s %s", key, val)
2256       if is_defined(pdfetcs.pgftextgs) then
2257         texsprint { "\\csname ", pdfetcs.pgftextgs, "\\endcsname{" , tr, "}" }
2258       elseif is_defined"TRP@list" then
2259         texsprint(catat11,{
2260           [[\if@files\immediate\write\@auxout{]],
2261           [[\string\g@addto@macro\string\TRP@list{]],
2262           tr,
2263           [[}]\fi]],
2264         })
2265         if not get_macro"TRP@list":find(tr) then
2266           texsprint(catat11,[[\global\TRP@reruntrue]])
2267         end
2268       else
2269         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")

```

```

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

```

2319 end

2320

Shading with *metafun* format.

2321 local function add_shading_resources (on, new)

2322 if new then

2323 local key, val = format("MPLibSh%s", on), format(pdfetcs.resfmt, on)

2324 if pdfmanagement then

2325 texsprint {

2326 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"

2327 }

2328 else

2329 local res = format("/%s %s", key, val)

2330 pdfetcs.fallback_update_resources("Shading",res,"@MPLibSh")

2331 end

2332 end

2333 end

2334 local function sh_pdfpageresources(shtype,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)

2335 for _,v in ipairs{ca,cb} do

2336 for i,vv in ipairs(v) do

2337 for ii,vvv in ipairs(vv) do

2338 v[ii][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv

2339 end

2340 end

2341 end

2342 local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"

2343 if steps > 1 then

2344 local list,bounds,encode = { },{ },{ }

2345 for i=1,steps do

2346 if i < steps then

2347 bounds[i] = format("%.3f", fractions[i] or 1)

2348 end

2349 encode[2*i-1] = 0

2350 encode[2*i] = 1

2351 os = fun2fmt:format(domain,tableconcat(ca[i],' '),tableconcat(cb[i],' '))

2352 :gsub(decimals,rmzeros)

2353 list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))

2354 end

2355 os = tableconcat {

2356 "<</FunctionType 3",

2357 format("/Bounds[%s]", tableconcat(bounds,' ')),

2358 format("/Encode[%s]", tableconcat(encode,' ')),

2359 format("/Functions[%s]", tableconcat(list,' ')),

2360 format("/Domain[%s]>>", domain),

2361 } :gsub(decimals,rmzeros)

2362 else

2363 os = fun2fmt:format(domain,tableconcat(ca[1],' '),tableconcat(cb[1],' '))

2364 :gsub(decimals,rmzeros)

2365 end

```

2366 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2367 os = tableconcat {
2368   format("</ShadingType %i", shtype),
2369   format("/ColorSpace %s", colorspace),
2370   format("/Function %s", objref),
2371   format("/Coords[%s]", coordinates),
2372   format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2373 } :gsub(decimals,rmzeros)
2374 local on, new = update_pdfobjs(os)
2375 add_shading_resources(on, new)
2376 return on
2377 end
2378
2379 local function get_mp_matrix (matrix)
2380   process("mplibtransformmatrix(%s);"):format(matrix), "@mplibtransformmatrix")
2381   return luamplib.transformmatrix
2382 end
2383
2384 local do_preobj_SH
2385 do
2386   pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2387     run_tex_code({
2388       [{"\color_model_new:nnn}],
2389       format("{mplibcolorspace_%s}", names:gsub(",","_")),
2390       format("{DeviceN}{names={%s}}", names),
2391       [{"\mplibtmptoks\expanded{ {\pdf_object_ref_last:} }"]],
2392     }, ccexplat)
2393     local colorspace = texgettoks"mplibtmptoks"
2394     t[names] = colorspace
2395     return colorspace
2396   end })
2397   local function color_normalize(ca,cb)
2398     if #cb == 1 then
2399       if #ca == 4 then
2400         cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2401       else -- #ca = 3
2402         cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2403       end
2404     elseif #cb == 3 then -- #ca == 4 : rgb to cmyk
2405       local c, m, y = 1-cb[1], 1-cb[2], 1-cb[3]
2406       local k = math.min(c, m, y)
2407       if k == 1 then
2408         c, m, y = 0, 0, 0
2409       else
2410         c, m, y = (c-k)/(1-k), (m-k)/(1-k), (y-k)/(1-k)
2411       end
2412       cb[1], cb[2], cb[3], cb[4] = c, m, y, k
2413     end
2414   end

```

```

2415 local function write_mesh_objs (shtype, colorspace, stream, devicen, perrow)
2416     local cc = colorspace == "/DeviceCMYK" and 4
2417             or colorspace == "/DeviceRGB" and 3
2418             or devicen or 1
2419     local dict = format(
2420         "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2421         shtype,
2422         perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2423         colorspace,
2424         (" 0 1"):rep(cc))
2425     local on, new
2426     if pdfmode then
2427         on, new = update_pdfobjs(dict, stream)
2428     else
2429         stream = format("%02X"):rep(stream:len()), stream:byte(1,stream:len())
2430         on, new = update_pdfobjs(dict, stream, true) -- hex is true
2431     end
2432     add_shading_resources(on, new)
2433     return on
2434 end
2435 local function colors_ff (colors)
2436     local t, devicen = { }, { }
2437     for i,v in ipairs(colors) do
2438         t[i] = { }
2439         for _,vv in ipairs(v) do
2440             local tt = tableconcat(vv," "):explode()
2441             for _,vvv in ipairs(tt) do
2442                 tableinsert(t[i], string.char(math.floor(vvv * 0xFF)))
2443             end
2444             devicen = #tt
2445         end
2446     end
2447     return t, devicen
2448 end
2449 local function coords_ffff (coords)
2450     local Xt, Yt = { }, { }
2451     for i,v in ipairs(coords) do
2452         for ii,vv in ipairs(v) do
2453             local t = ii % 2 == 1 and Xt or Yt
2454             t[#t+1] = tonumber(vv)
2455         end
2456     end
2457     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2458     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2459     local wd, ht = xmax - xmin, ymax - ymin
2460     for i,v in ipairs(coords) do
2461         for ii,vv in ipairs(v) do
2462             local min = ii % 2 == 1 and xmin or ymin
2463             local dim = ii % 2 == 1 and wd or ht

```



```

2464     coords[i][ii] = string.pack(">H", math.floor((vv - min)/dim * 0xFFFF ))
2465     end
2466     end
2467     return coords, xmin, ymin, wd, ht
2468 end
2469 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2470     local perrow = prescript.sh_lattice_perrow or 2
2471     local steps = tonumber(prescript.sh_step) or 0
2472     local coords, colors = { }, { }
2473     if steps < 4 then
2474         local path = object.path
2475         for _,i in ipairs{1,2,4,3} do
2476             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2477         end
2478         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}}, {{1,1,0}} }
2479         colorspace = "/DeviceRGB"
2480     else
2481         local t = prescript.sh_lattice_data:explode()
2482         for i = 1, #t, 2 do
2483             coords[#coords+1] = { t[i], t[i+1] }
2484         end
2485         for i = 1, steps do
2486             if not ca[i] then err"colorspace mismatch?" end
2487             colors[#colors+1] = { ca[i] }
2488         end
2489     end
2490     if #coords % perrow ~= 0 then err"vertices_per_row mismatches lattice_coords" end
2491
2492     local stream, xmin, ymin, wd, ht, devicen = { }
2493     coords, xmin, ymin, wd, ht = coords_ffff(coords)
2494     colors, devicen = colors_ff(colors)
2495     for i = 1, #coords do
2496         stream[#stream+1] = tableconcat(coords[i])
2497         stream[#stream+1] = tableconcat(colors[i])
2498     end
2499
2500     local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2501     local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2502     return on, matrix
2503 end
2504 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2505     local steps = tonumber(prescript.sh_step) or 0
2506     local coords, colors, edges = { }, { }, { }
2507     if steps < 3 then
2508         local path = object.path
2509         for i = 1, 3 do
2510             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2511         end
2512         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}} }

```

```

2513     edges = { 0, 0, 0 }
2514     colorspace = "/DeviceRGB"
2515 else
2516     for i = 1, steps do
2517         local t = prescript["sh_triangle_vertex_" .. i]:explode()
2518         if #t == 2 then
2519             coords[#coords+1] = { t[1], t[2] }
2520         elseif #t == 6 then
2521             coords[#coords+1] = { t[1], t[2] }
2522             coords[#coords+1] = { t[3], t[4] }
2523             coords[#coords+1] = { t[5], t[6] }
2524         end
2525         if not ca[i] then err"colorspace mismatch?" end
2526         colors[#colors+1] = { ca[i] }
2527         edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])
2528     end
2529 end
2530
2531 local stream, xmin, ymin, wd, ht, devicen = { }
2532 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2533 colors, devicen = colors_ff(colors)
2534 for i = 1, #coords do
2535     stream[#stream+1] = string.char(edges[i])
2536     stream[#stream+1] = tableconcat(coords[i])
2537     stream[#stream+1] = tableconcat(colors[i])
2538 end
2539
2540 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2541 local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2542 return on, matrix
2543 end
2544 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2545     local tensor = prescript.sh_type == "tensor"
2546     local steps = tonumber(prescript.sh_step) or 0
2547     local coords, colors, edges = { }, { }, { }
2548     if steps < 2 then
2549         local path, t = object.path, { }
2550         for i = 1, 4 do
2551             t[#t+1] = path[i].x_coord
2552             t[#t+1] = path[i].y_coord
2553             t[#t+1] = path[i].right_x
2554             t[#t+1] = path[i].right_y
2555             local j = i == 4 and 1 or i+1
2556             t[#t+1] = path[j].left_x
2557             t[#t+1] = path[j].left_y
2558         end
2559         if tensor then
2560             t = table.move(prescript.sh_tensor_path:explode(), 1, 8, #t+1, t)
2561         end

```

```

2562     coords = { t }
2563     colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2564     edges = { 0 }
2565     colorspace = "/DeviceRGB"
2566   else
2567     for i = 1, steps do
2568       local edge = tonumber(prescript["sh_coons_edge_"..i])
2569       if edge then
2570         edges[#edges+1] = edge
2571         coords[#coords+1] = prescript["sh_coons_path_"..i]:explode()
2572         if edge == 0 then
2573           if not (ca[i] and cb[i] and ca[i+1] and cb[i+1]) then err"colorspace mismatch?" end
2574           colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2575         else
2576           if not (ca[i] and cb[i]) then err"colorspace mismatch?" end
2577           colors[#colors+1] = { ca[i], cb[i] }
2578         end
2579       end
2580     end
2581   end
2582
2583   local stream, xmin, ymin, wd, ht, devicen = { }
2584   coords, xmin, ymin, wd, ht = coords_ffff(coords)
2585   colors, devicen = colors_ff(colors)
2586   for i = 1, #coords do
2587     stream[#stream+1] = string.char(edges[i])
2588     stream[#stream+1] = tableconcat(coords[i])
2589     stream[#stream+1] = tableconcat(colors[i])
2590   end
2591
2592   local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin):gsub(decimals,rmzeros)
2593   local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2594   return on, matrix
2595 end
2596 function do_preobj_SH(object, prescript)
2597   local shade_no
2598   local sh_type = prescript and prescript.sh_type
2599   if not sh_type then return end
2600
2601   local domain = prescript.sh_domain or "0 1"
2602   local centera = (prescript.sh_center_a or "0 0"):explode()
2603   local centerb = (prescript.sh_center_b or "0 0"):explode()
2604   local transform = prescript.sh_transform == "yes"
2605   local sx,sy,sr,dx,dy = 1,1,1,0,0
2606   if transform then
2607     local first = (prescript.sh_first or "0 0"):explode()
2608     local setx = (prescript.sh_set_x or "0 0"):explode()
2609     local sety = (prescript.sh_set_y or "0 0"):explode()
2610     local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0

```

```

2611     if x ~= 0 and y ~= 0 then
2612         local path = object.path
2613         local path1x = path[1].x_coord
2614         local path1y = path[1].y_coord
2615         local path2x = path[x].x_coord
2616         local path2y = path[y].y_coord
2617         local dxa = path2x - path1x
2618         local dya = path2y - path1y
2619         local dx = setx[2] - first[1]
2620         local dy = sety[2] - first[2]
2621         if dxa ~= 0 and dya ~= 0 and dx ~= 0 and dy ~= 0 then
2622             sx = dxa / dx ; if sx < 0 then sx = - sx end
2623             sy = dya / dy ; if sy < 0 then sy = - sy end
2624             sr = math.sqrt(sx^2 + sy^2)
2625             dx = path1x - sx*first[1]
2626             dy = path1y - sy*first[2]
2627         end
2628     end
2629 end
2630 local ca, cb, colorspace, steps, fractions
2631 ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode":" }
2632 cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode":" }
2633 steps = tonumber(prescript.sh_step) or 1
2634 if steps > 1 then
2635     fractions = { prescript.sh_fraction_1 or 0 }
2636     for i=2,steps do
2637         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2638         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode":"
2639         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode":"
2640     end
2641 end
2642 if prescript.mplib_spotcolor then
2643     ca, cb = { }, { }
2644     local names, pos, objref = { }, -1, ""
2645     local script = object.prescript:explode"\13+"
2646     for i=#script,1,-1 do
2647         local name, value
2648         if script[i]:find"mplib_spotcolor" then
2649             local t = script[i]:explode"="[2]:explode":"
2650             value, objref, name = t[1], t[2], t[3]
2651         elseif script[i]:find"mplib_texcolor" then
2652             local t = script[i]:explode"="[2]:explode"!"
2653             name, value = t[1], (t[2] or 100)/100
2654         end
2655         if name and value then
2656             if not names[name] then
2657                 pos = pos+1
2658                 names[name] = pos
2659                 names[#names+1] = name

```

```

2660         end
2661         local t = { }
2662         for j=1,names[name] do t[#t+1] = 0 end
2663         t[#t+1] = value
2664         tableinsert(#ca == #cb and ca or cb, t)
2665     end
2666 end
2667 for _,t in ipairs{ca,cb} do
2668     for _,tt in ipairs(t) do
2669         for i=1,#names-#tt do tt[#tt+1] = 0 end
2670     end
2671 end
2672 if #names == 1 then
2673     colorspace = objref
2674 else
2675     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2676 end
2677 else
2678     local model = 0
2679     for _,t in ipairs{ca,cb} do
2680         for _,tt in ipairs(t) do
2681             model = model > #tt and model or #tt
2682         end
2683     end
2684     for _,t in ipairs{ca,cb} do
2685         for _,tt in ipairs(t) do
2686             if #tt < model then
2687                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2688             end
2689         end
2690     end
2691     colorspace = model == 4 and "/DeviceCMYK"
2692                 or model == 3 and "/DeviceRGB"
2693                 or model == 1 and "/DeviceGray"
2694                 or err"unknown color model"
2695 end
2696 local extend = prescript.sh_extend
2697 local mesh_matrix
2698 if sh_type == "linear" then
2699     local coordinates = format("%f %f %f %f",
2700         dx + sx*centera[1], dy + sy*centera[2],
2701         dx + sx*centerb[1], dy + sy*centerb[2])
2702     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2703 elseif sh_type == "circular" then
2704     local factor = prescript.sh_factor or 1
2705     local radiusa = factor * prescript.sh_radius_a
2706     local radiusb = factor * prescript.sh_radius_b
2707     local coordinates = format("%f %f %f %f %f %f",
2708         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,

```

```

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

```

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

```

2737 local function add_pattern_resources (key, val)
2738     if pdfmanagement then
2739         texsprint {
2740             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2741         }
2742     else
2743         local res = format("/%s %s", key, val)
2744         if is_defined(pdfetcs.pgfpattern) then
2745             texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2746         else
2747             pdfetcs.fallback_update_resources("Pattern",res,"@MPlibPt")
2748         end
2749     end
2750 end
2751
2752 if pdfmode then
2753     function luamplib.dolatelua (on, os, matrix)
2754         local h, v = pdf.getpos()
2755         local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2756         matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2757     end
2758 end

```

```

2756      :gsub(decimals,rmzeros)
2757      pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2758      pdf.refobj(on)
2759  end
2760 else
2761  pdfetcs.shadingpatterns = { }
2762  pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2763  function luamplib.dolatelua (on, kind, xobj)
2764    local h, v
2765    local t = pdfetcs.shadingpatterns[on] or { }
2766    local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2767                  or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2768    if shift then
2769      h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2770    else
2771      h, v = pdf.getpos()
2772      h = format("%f", h/factor) :gsub(decimals,rmzeros)
2773      v = format("%f", v/factor) :gsub(decimals,rmzeros)
2774    end
2775    if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2776      warn"Rerun to get correct shading pattern"
2777    end
2778    local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2779    local init = pdfetcs.shadingpatterninit_w
2780    if init then pdfetcs.shadingpatterninit_w = nil end
2781    local f = ioopen(name, init and "w" or "a")
2782    if f then
2783      f:write(("<< %s %s %s\n"):format(on, h, v))
2784      f:close()
2785    else
2786      err"cannot write a file. check the cache dir path"
2787    end
2788  end
2789 end
2790 local function do_preobj_shading (object, prescript)
2791   if not prescript or not prescript.sh_operand_type then return end
2792   local on,_,matrix = do_preobj_SH(object, prescript)
2793   local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2794   matrix = matrix or "1 0 0 1 0 0"
2795   if prescript.sh_in_xobj == "yes" then
2796     on = update_pdfobjs(("<<%s/Matrix[%s]>>"):format(os, matrix))
2797     goto skip_latelua
2798   end
2799   on = update_pdfobjs()
2800   if pdfmode then
2801     put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"]],[",matrix,"]])"})
2802   else
2803     local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2804                  or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}

```

```

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

```

This seems to be needed for proper functioning:

```

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

2818 local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2819 local mt = matrix:explode()
2820 matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2821 texsprint{ "\special{pdf:put ", format(pdfetcs.resfmt, on),
2822   format(" <<%s/Matrix[%s]>>", os, matrix) }
2823 put2output("\latelua{ luamplib.dolatelua(%s,%s) }", on,
2824   xobj and ("%s',[[%s]]"):format(xobj[1], xobj[2]))
2825 end
2826 ::skip_latelua::
2827 local key, val = format("MPlibPt%s", on), format(pdfetcs.resfmt, on)
2828 add_pattern_resources(key,val)

```

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

```

2829 local stroke = prescript.sh_stroking
2830 if stroke == "filldraw" or stroke == "both" then
2831   pdf_literalcode("/Pattern cs/%s scn /Pattern CS/%s SCN", key, key)
2832 elseif stroke == "draw" or stroke == "yes" then
2833   pdf_literalcode("/Pattern CS/%s SCN", key)
2834 else
2835   pdf_literalcode("/Pattern cs/%s scn", key)
2836 end

```

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

```

2837 prescript.sh_type = nil
2838 end
2839

```

Tiling Patterns

```

2840 pdfetcs.patterns = { }
2841 local function gather_resources (optres, ispattern)

```



```

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

```

We still have to deal with Shading resources.

```

2885 if luatexbase.callbacktypes.finish_pdffile then
2886   if pdfetcs.Shading_res then
2887     t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2888   end

```

```

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

```

2898         warn"transparent package is not fully functional without pdfmanagement code."
2899     end
2900     if luatexbase.callbacktypes.finish_pdffile then
2901         for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2902             local tt = pdfetcs[v.."_res"]
2903             if tt then
2904                 t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2905             end
2906         end
2907     else
2908         local res = pdfetcs.getpageres()
2909         if res then
2910             t[#t+1] = res
2911         end
2912     end
2913 end
2914 local result = tableconcat(t)
2915 if optres ~= "" then
2916     for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2917         local res = optres:match("/"..v.."s*%b<>")
2918         if res then
2919             if result:find("/"..v) then
2920                 res = res:match("<<(.+)>>$")
2921                 result = result:gsub("/"..v.."s*<<", "%1"..res, 1)
2922             else
2923                 result = result .. res
2924             end
2925         end
2926     end
2927 end
2928 return result
2929 end
2930 function luamplib.registerpattern ( boxid, name, opts )
2931     local box = texgetbox(boxid)
2932     local wd = format("%.3f",box.width/factor)
2933     local hd = format("%.3f",(box.height+box.depth)/factor)
2934     info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))

```

```

2935 if opts.xstep == 0 then opts.xstep = nil end
2936 if opts.ystep == 0 then opts.ystep = nil end
2937 if opts.colored == nil then
2938     opts.colored = opts.coloured
2939     if opts.colored == nil then
2940         opts.colored = true
2941     end
2942 end
2943 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2944 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2945 if opts.matrix and opts.matrix:find"%a" then
2946     local t = get_mp_matrix(opts.matrix)
2947     opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2948     opts.xshift = opts.xshift or format("%f", t[5])
2949     opts.yshift = opts.yshift or format("%f", t[6])
2950 end
2951 local attr = {
2952     "/Type/Pattern",
2953     "/PatternType 1",
2954     format("/PaintType %i", opts.colored and 1 or 2),
2955     "/TilingType 2",
2956     format("/XStep %s", opts.xstep or wd),
2957     format("/YStep %s", opts.ystep or hd),
2958     format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2959 }
2960 local optres = opts.resources or ""
2961 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2962 local patterns = pdfetcs.patterns
2963 if pdfmode then
2964     if opts.bbox then
2965         attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2966     end
2967     attr = tableconcat(attr) :gsub(decimals, rmzeros)
2968     local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2969     patterns[name] = { id = index, colored = opts.colored }
2970 else
2971     local cnt = #patterns + 1
2972     local objname = "@mplibpattern" .. cnt
2973     local metric = format("bbox %s", opts.bbox or format("0 0 %s %s", wd, hd))
2974     texsprint {
2975         "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2976         "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2977         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2978         "\\special{pdf:bcontent}",
2979         "\\special{pdf:bxobj ", objname, " ", metric, "}",
2980         "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2981         "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2982         "\\special{pdf:put @resources <<", optres, ">>}",
2983         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",

```

```

2984     "\\special{pdf:econtent}}",
2985   }
2986   patterns[cnt] = objname
2987   patterns[name] = { id = cnt, colored = opts.colored }
2988   patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
2989 end
2990 end
2991
2992 local do_preobj_PAT
2993 do
2994   local function pattern_colorspace (cs)
2995     local on, new = update_pdfobjs(format("/Pattern %s]", cs))
2996     if new then
2997       local key, val = format("MPlibCS%i", on), format(pdfetcs.resfmt, on)
2998       if pdfmanagement then
2999         texsprint {
3000           "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{" , val, "}"
3001         }
3002       else
3003         local res = format("/%s %s", key, val)
3004         if is_defined(pdfetcs.pgfcolorspace) then
3005           texsprint { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{" , res, "}" }
3006         else
3007           pdfetcs.fallback_update_resources("ColorSpace", res, "@MPlibCS")
3008         end
3009       end
3010     end
3011     return on
3012   end
3013   local function uncoloredGS(color, key, fill)
3014     local cs
3015     if color:find" cs " then
3016       local name
3017       if fill then
3018         name, color = color:match"^/(.-) cs (.-) sc"
3019       else
3020         name, color = color:match" /(-) CS (-) SC"
3021       end
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({ "\\mplibtmptoks\\expanded{\\pdf_object_ref:n{" , name, "}}}" }, ccexplat)
3029         cs = texgettoks"mplibtmptoks"
3030       end
3031     else
3032       if fill then

```

```

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

```

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

```

3064   if stroke == "filldraw" or stroke == "both" then
3065     pdf_literalcode("%s %s", fillgs, strkgs)
3066   elseif stroke == "draw" or stroke == "yes" then
3067     pdf_literalcode(strkgs)
3068   else
3069     pdf_literalcode(fillgs)
3070   end
3071   if not patt.done then
3072     local val = pdfmode and format("%s 0 R",index) or patterns[index]
3073     add_pattern_resources(key,val)
3074   end
3075   patt.done = true
3076 end
3077 end
3078

```

Fading

```

3079 pdfetcs.fading = { }
3080 local function do_preobj_FADE (object, prescript)
3081   local fd_type = prescript and prescript.mplibfadetype
3082   local fd_stop = prescript and prescript.mplibfadestate
3083   if not fd_type then
3084     return fd_stop -- returns "stop" (if picture) or nil
3085   end
3086   local on, os, new
3087   if fd_type == "masking" then
3088     local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3089     on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
3090     local bc = prescript.mplibmaskingbgcolor
3091     bc = bc and bc:gsub(":", " ")
3092     bc = bc and ("BC[%s]"):format(bc):gsub(decimals, rmzeros) or ""
3093     os = format("<</SMask<</S/Luminosity/G %s%>>>",
3094                 pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3095   else
3096     local bbox = prescript.mplibfadebbox:explode":"
3097     local dx, dy = -bbox[1], -bbox[2]
3098     local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
3099     if not vec then
3100       if fd_type == "linear" then
3101         vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3102       else
3103         local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3104         vec = {centerx, centery, centerx, centery} -- center for both circles
3105       end
3106     end
3107     local coords = { vec[1], vec[2], vec[3], vec[4] }
3108     if fd_type == "linear" then
3109       coords = format("%f %f %f %f", tableunpack(coords))
3110     elseif fd_type == "circular" then
3111       local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3112       local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":"
3113       tableinsert(coords, 3, radius[1])
3114       tableinsert(coords, radius[2])
3115       coords = format("%f %f %f %f %f %f", tableunpack(coords))
3116     else
3117       err("unknown fading method '%s'", fd_type)
3118     end
3119     fd_type = fd_type == "linear" and 2 or 3
3120     local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3121     local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":" }
3122     local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":" }
3123     if steps > 1 then
3124       fractions = { prescript.sh_fraction_1 or 0 }
3125       for i=2, steps do
3126         fractions[i] = prescript[format("sh_fraction_%i", i)] or (i/steps)

```

```

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

```

Transparency Group

```

3164 pdfetcs.tr_group = { shifts = { } }
3165 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3166 local function do_preobj_GRP (object, prescript)
3167     local grstate = prescript and prescript.gr_state
3168     if not grstate then return end
3169     local trgroup = pdfetcs.tr_group
3170     if grstate == "start" then
3171         trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3172         trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3173         trgroup.wrapped = false

```

```

3174   for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode","+") do
3175       trgroup[v] = true
3176   end
3177   trgroup.bbox = prescript.mplibgroupbbox:explode":"
3178   put2output[[\begingroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3179 elseif grstate == "stop" then
3180     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3181     put2output(tableconcat{
3182         "\\egroup",
3183         format("\\wd\mplibscratchbox %fbp", urx-llx),
3184         format("\\ht\mplibscratchbox %fbp", ury-lly),
3185         "\\dp\mplibscratchbox 0pt",
3186     })
3187     local grattr
3188     if trgroup.off then
3189         grattr = ""
3190     else
3191         local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>",
3192             trgroup.isolated, trgroup.knockout))
3193         grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3194     end
3195     local res = gather_resources("")
3196     local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3197     if pdfmode then
3198         if trgroup.wrapped then
3199             put2output(tableconcat{
3200                 "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3201                 "/BBox[" .. bbox .. "] resources{" .. res .. "}}\\mplibscratchbox",
3202                 "\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}",
3203             })
3204         end
3205         put2output(tableconcat{
3206             "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3207             "/BBox[" .. bbox .. "]", grattr, "} resources{" .. res .. "}}\\mplibscratchbox",
3208             "\\luamplibtagasgroupput{" .. trgroup.name .. "}",
3209             [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
3210             [[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]],
3211             [[\box\mplibscratchbox]],
3212             "\\endgroup",
3213             "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{",
3214             "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3215             "\\useboxresource \\the\\lastsavedboxresourceindex",
3216             "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3217             "\\box\\mplibscratchbox}",
3218         })
3219     else
3220         if trgroup.wrapped then
3221             trgroup.cnt = (trgroup.cnt or 0) + 1
3222             local objname = format("@mplibtrgr%s", trgroup.cnt)

```



```

3223     put2output(tableconcat{
3224         "\\special{pdf:boxobj ", objname, " bbox ", bbox, "}",
3225         "\\unhbox\\mplibscratchbox",
3226         "\\special{pdf:put @resources <<", res, ">>}",
3227         "\\special{pdf:exobj}",
3228         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3229     })
3230 end
3231 trgroup.cnt = (trgroup.cnt or 0) + 1
3232 local objname = format("@mplibtrgr%s", trgroup.cnt)
3233 put2output(tableconcat{
3234     "\\special{pdf:boxobj ", objname, " bbox ", bbox, "}",
3235     "\\unhbox\\mplibscratchbox",
3236     "\\special{pdf:put @resources <<", res, ">>}",
3237     "\\special{pdf:exobj <<", grattr, ">>}",
3238     "\\luamplibtagasgroupput{",trgroup.name,"}",
3239     "\\special{pdf:uxobj ", objname, "}",
3240     "}\\endgroup",
3241 })
3242 token.set_macro("luamplib.group."..trgroup.name, tableconcat{
3243     "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3244     "\\special{pdf:uxobj ", objname, "}",
3245     "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3246     "\\box\\mplibscratchbox",
3247     }, "global")
3248 end
3249 trgroup.shifts[trgroup.name] = { llx, lly }
3250 end
3251 return grstate
3252 end
3253 function luamplib.registergroup (boxid, name, opts)
3254     if opts.asgroup and opts.asgroup:find"wrapped" then
3255         luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3256         run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
3257         opts.asgroup = opts.asgroup:gsub("wrapped","")
3258     end
3259     local box = texgetbox(boxid)
3260     local wd, ht, dp = node.getwhd(box)
3261     local is_mask = opts.asgroup and opts.asgroup:find"masking"
3262     local res = opts.resources or ""
3263     res = gather_resources(res)
3264     local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3265     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
3266     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
3267     if opts.matrix and opts.matrix:find"%a" then
3268         local t = get_mp_matrix(opts.matrix)
3269         opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3270     end
3271     local grtype = 3

```

```

3272 if opts.bbox then
3273   attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3274   grtype = 2
3275 end
3276 local mplx, mply = get_macro'MPlx', get_macro'MPly'
3277 if is_mask then
3278   local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3279   t[5], t[6] = t[5]+mplx, t[6]+mply
3280   opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3281   mplx, mply = 0, 0
3282 end
3283 if opts.matrix then
3284   attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3285   grtype = opts.bbox and 4 or 1
3286 end
3287 if opts.asgroup and not opts.asgroup:find"off" then
3288   local t = { isolated = false, knockout = false, masking = false }
3289   for _,v in ipairs(opts.asgroup:gsub("%s",""):explode",+") do t[v] = true end
3290   local on
3291   if t.masking then
3292     on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorspace or "/DeviceGray"))
3293   else
3294     local cs = opts.colorspace and ("/CS%s"):format(opts.colorspace) or ""
3295     on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3296   end
3297   attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3298 end
3299 local trgroup = pdfetcs.tr_group
3300 trgroup.shifts[name] = { mplx, mply }
3301 local whd
3302 if pdfmode then
3303   attr = tableconcat(attr) :gsub(decimals,rmzeros)
3304   local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3305   token.set_macro("luamplib.group."..name, tableconcat{
3306     "\\useboxresource ", index,
3307     }, "global")
3308   whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3309 else
3310   trgroup.cnt = (trgroup.cnt or 0) + 1
3311   local objname = format("@mplibtrgr%s", trgroup.cnt)
3312   texsprint {
3313     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3314     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3315     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3316     "\\special{pdf:bcontent}",
3317     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3318     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3319     "\\special{pdf:put @resources <<", res, ">>}",
3320     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",

```

```

3321     "\\special{pdf:econtent}}",
3322   }
3323   token.set_macro("luamplib.group"..name, tableconcat{
3324     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3325     "\\wd\\mplibscratchbox ", wd, "sp",
3326     "\\ht\\mplibscratchbox ", ht, "sp",
3327     "\\dp\\mplibscratchbox ", dp, "sp",
3328     "\\box\\mplibscratchbox",
3329   }, "global")
3330   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3331 end
3332 info("w/h/d of group '%s': %s", name, whd)
3333 end
3334

```

luamplib.convert: flushing figures

```

3335 do
3336   local function stop_special_effects(fade,opaq)
3337     if fade then -- fading
3338       stop_pdf_code()
3339     end
3340     if opaq then -- opacity
3341       pdf_literalcode(opaq)
3342     end
3343   end
3344

```

For parsing prescript materials.

```

3345   local function script2table(s)
3346     local t = {}
3347     for _,i in ipairs(s:explode("\\13+")) do
3348       local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3349       if k and v and k ~= "" and not t[k] then
3350         t[k] = v
3351       end
3352     end
3353     return t
3354   end
3355

```

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

```

3356   local function pdf_textfigure(font,size,text,width,height,depth)
3357     text = text:gsub(".",function(c)
3358       return format("\\hbox{\\char%i}",string.byte(c)) -- kerning happens in metapost : false
3359     end)
3360     put2output("\\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
3361   end
3362
3363   local bend_tolerance = 131/65536

```

```

3364
3365 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3366
3367 local function pen_characteristics(object)
3368     local t = mplib.pen_info(object)
3369     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3370     divider = sx*sy - rx*ry
3371     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3372 end
3373
3374 local function concat(px, py) -- no tx, ty here
3375     return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
3376 end
3377
3378 local function curved(ith,pth)
3379     local d = pth.left_x - ith.right_x
3380     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3381         abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3382         d = pth.left_y - ith.right_y
3383         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3384             abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3385             return false
3386         end
3387     end
3388     return true
3389 end
3390
3391 local function flushnormalpath(path,open)
3392     local pth, ith
3393     for i=1,#path do
3394         pth = path[i]
3395         if not ith then
3396             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3397         elseif curved(ith,pth) then
3398             pdf_literalcode("%f %f %f %f %f c",
3399                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3400         else
3401             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3402         end
3403         ith = pth
3404     end
3405     if not open then
3406         local one = path[1]
3407         if curved(pth,one) then
3408             pdf_literalcode("%f %f %f %f %f %f c",
3409                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3410         else
3411             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3412         end

```

```

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

```

Finally, flush figures by inserting PDF literals.

```

3450 local function flush (result,flusher)
3451     if result then
3452         local figures = result.fig
3453         if figures then
3454             for f=1, #figures do
3455                 info("flushing figure %s",f)
3456                 local figure = figures[f]
3457                 local objects = figure:objects()
3458                 local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3459                 local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false

```

```

3460     local bbox = figure:boundingbox()
3461     local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3462     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()

3463     else

```

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

```

3464     if tex_code_pre_mplib[f] then
3465         put2output(tex_code_pre_mplib[f])
3466     end
3467     pdf_startfigure(fignum,llx,lly,urx,ury)
3468     start_pdf_code()
3469     if objects then
3470         local savedpath = nil
3471         local savedhtap = nil
3472         for o=1,#objects do
3473             local object      = objects[o]
3474             local objecttype   = object.type

```

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

```

3475         local prescript      = object.prescript
3476         prescript = prescript and script2table(prescript) -- prescript is now a table
3477         do_preobj_CR(object,prescript) -- color
3478         local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3479         local fading_ = do_preobj_FADE(object,prescript) -- fading
3480         do_preobj_PAT(object,prescript) -- tiling pattern
3481         do_preobj_shading(object,prescript) -- shading pattern
3482         local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3483         if prescript and prescript.mplibtexboxid then
3484             put_tex_boxes(object,prescript)
3485         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3486         elseif objecttype == "start_clip" then
3487             local evenodd = not object.istext and object.postscript == "evenodd"
3488             start_pdf_code()
3489             flushnormalpath(object.path,false)
3490             pdf_literalcode(evenodd and "W* n" or "W n")
3491         elseif objecttype == "stop_clip" then
3492             stop_pdf_code()
3493             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3494         elseif objecttype == "special" then

```

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

```

3495         if prescript and prescript.postmplibverbtex then
3496             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtex

```

```

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

```

Removed from ConT_EXt general: color stuff.

```

3528         local ml = object.miterlimit
3529         if ml and ml ~= miterlimit then
3530             miterlimit = ml
3531             pdf_literalcode("%f M",ml)
3532         end
3533         local lj = object.linejoin
3534         if lj and lj ~= linejoin then
3535             linejoin = lj
3536             pdf_literalcode("%i j",lj)
3537         end
3538         local lc = object.linecap
3539         if lc and lc ~= linecap then
3540             linecap = lc
3541             pdf_literalcode("%i J",lc)
3542         end
3543         local dl = object.dash

```

```

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

```

Added : shading

```

3569     local shade_no, shade_stroke, shade_cm = do_preobj_SH(object,prescript) -- shading
3570     if shade_no then
3571         pdf_literalcode"q /Pattern cs"
3572         objecttype = false
3573     end
3574     if transformed then
3575         start_pdf_code()
3576     end
3577     if path then
3578         if savedpath then
3579             for i=1,#savedpath do
3580                 local path = savedpath[i]
3581                 if transformed then
3582                     flushconcatpath(path,open)
3583                 else
3584                     flushnormalpath(path,open)
3585                 end
3586             end
3587             savedpath = nil
3588         end
3589         if transformed then
3590             flushconcatpath(path,open)

```



```

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

```

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

```

3610     if path then
3611         if transformed then
3612             start_pdf_code()
3613         end
3614         if savedhtap then
3615             for i=1,#savedhtap do
3616                 local path = savedhtap[i]
3617                 if transformed then
3618                     flushconcatpath(path,open)
3619                 else
3620                     flushnormalpath(path,open)
3621                 end
3622             end
3623             savedhtap = nil
3624             evenodd = true
3625         end
3626         if transformed then
3627             flushconcatpath(path,open)
3628         else
3629             flushnormalpath(path,open)
3630         end
3631         if objecttype == "fill" then
3632             pdf_literalcode(evenodd and "h f*" or "h f")
3633         elseif objecttype == "outline" then
3634             pdf_literalcode(open and "S" or "h S")
3635         elseif objecttype == "both" then
3636             pdf_literalcode(evenodd and "h B*" or "h B")
3637         end

```

```

3638             if transformed then
3639                 stop_pdf_code()
3640             end
3641         end

```

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

```

3642             if shade_no then -- shading
3643                 pdf_literalcode("W%s %s %s/MPLibSh%s sh Q",
3644                     evenodd and "*" or "",
3645                     shade_stroke and "s" or "n",
3646                     shade_cm and shade_cm.." cm " or "",
3647                     shade_no)
3648             end
3649         end
3650     end
3651     if fading_ == "start" then
3652         pdfetcs.fading.specialeffects = {fading_, tr_opaq}
3653     elseif trgroup == "start" then
3654         pdfetcs.tr_group.specialeffects = {fading_, tr_opaq}
3655     elseif fading_ == "stop" then
3656         local se = pdfetcs.fading.specialeffects
3657         if se then stop_special_effects(se[1], se[2]) end
3658     elseif trgroup == "stop" then
3659         local se = pdfetcs.tr_group.specialeffects
3660         if se then stop_special_effects(se[1], se[2]) end
3661     else
3662         stop_special_effects(fading_, tr_opaq)
3663     end
3664     if fading_ or trgroup then -- extgs resetted
3665         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3666     end
3667 end
3668 end
3669 stop_pdf_code()
3670 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimt_Ex code.

```

3671     for _,v in ipairs(figcontents) do
3672         if type(v) == "table" then
3673             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3674         else
3675             texsprint(v)
3676         end
3677     end
3678     if #figcontents.post > 0 then texsprint(figcontents.post) end
3679     figcontents = { post = { } }
3680 end
3681 end
3682 end
3683 end

```

```

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

```

2.2 $\TeX$ package

First we need to load some packages.

```

3705 \ifcsname ProvidesPackage\endcsname

```

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

```

3706 \NeedsTeXFormat{LaTeX2e}
3707 \ProvidesPackage{luamplib}
3708   [2026/08/12 v2.42.7 mplib package for LuaTeX]
3709 \fi
3710 \ifdefined\newluafunction\else
3711   \input ltluatex
3712 \fi

```

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

```

3713 \ifnum\outputmode=0
3714   \ifdefined\AddToHookNext
3715     \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3716     \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3717     \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3718   \else
3719     \input atbegshi.sty
3720     \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3721     \let\luamplibatfirstshipout\AtBeginShipoutFirst

```

```

3722 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3723 \fi
3724 \fi

```

Loading of lua code.

```

3725 \directlua{require("luamplib")}

```

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

```

3726 \ifx\pdfoutput\undefined
3727 \let\pdfoutput\outputmode
3728 \fi
3729 \ifx\pdfliteral\undefined
3730 \protected\def\pdfliteral{\pdfextension literal}
3731 \fi

```

Set the format for METAPOST.

```

3732 \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.

```

3733 \ifnum\pdfoutput>0
3734 \let\mplibtoPDF\pdfliteral
3735 \else
3736 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3737 \ifcsname PackageInfo\endcsname
3738 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3739 \else
3740 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3741 \fi
3742 \fi

```

To make mplibcode typeset always in horizontal mode.

```

3743 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3744 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3745 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

3746 \def\mplibsetupcatcodes{%
3747 %catcode`\{=12 %catcode`\}=12
3748 \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3749 \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^M=12
3750 }

```

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

```

3751 \ifnum\outputmode>0 %
3752 \def\mplibputtextbox#1#2#3#4{%
3753 \pdfextension save\relax
3754 \vbox to 0pt{\vss
3755 \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
3756 \kern#3bp}%
3757 \pdfextension restore\relax}

```

```

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

      use Transparency Group

3764 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3765 \def\usemplibgroupmain#1{%
3766 \prependtomplibbox\hbox dir TLT\bgroup
3767 \csname luamplib.group.#1\endcsname
3768 \egroup
3769 }
3770 \protected\def\mplibgroup#1{%
3771 \begingroup
3772 \def\MPllx{0}\def\MPlly{0}%
3773 \def\mplibgroupname{#1}%
3774 \mplibgroupgetnexttok
3775 }
3776 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3777 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
3778 \def\mplibgroupbranch{%
3779 \ifx [\nexttok
3780 \expandafter\mplibgroupopts
3781 \else
3782 \ifx\mplibsptoken\nexttok
3783 \expandafter\expandafter\expandafter\mplibgroupskipspace
3784 \else
3785 \let\mplibgroupoptions\empty
3786 \expandafter\expandafter\expandafter\mplibgroupmain
3787 \fi
3788 \fi
3789 }
3790 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3791 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3792 \protected\def\endmplibgroup{\egroup
3793 \directlua{ luamplib.registergroup(
3794 \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3795 )}}%
3796 \endgroup
3797 }

      Patterns

3798 {\def\:{\global\let\mplibsptoken= } \: }
3799 \protected\def\mplipattern#1{%
3800 \begingroup
3801 \def\MPllx{0}\def\MPlly{0}%
3802 \def\mplibpatternname{#1}%
3803 \mplibpatterngetnexttok

```

```

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

```

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

```

3833 \def\mpfiginstancename{@mpfig}
3834 \protected\def\mpfig{%
3835   \begingroup
3836   \futurelet\nexttok\mplibmpfigbranch
3837 }
3838 \def\mplibmpfigbranch{%
3839   \ifx *\nexttok
3840     \expandafter\mplibprempfig
3841   \else
3842     \ifx [\nexttok
3843       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3844     \else
3845       \expandafter\expandafter\expandafter\mplibmainmpfig
3846     \fi
3847   \fi
3848 }
3849 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
3850 \def\mplibmainmpfig{%

```

```

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

```

The Plain-specific stuff.

```

3895 \unless\ifcsname ver@luamplib.sty\endcsname
3896 \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3897 \protected\def\mplibcode{%

```

```

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

```

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

```

3921 \newenvironment{mplibcode}[1][{}%
3922 \xdef\currentmpinstancename{#1}%
3923 \mplibtmptoks{}\ltxdomplibcode
3924 }{}
3925 \def\ltxdomplibcode{%
3926 \begingroup
3927 \mplibsetupcatcodes
3928 \ltxdomplibcodeindeed
3929 }
3930 \def\mplib@mplibcode{mplibcode}
3931 \long\def\ltxdomplibcodeindeed#1\end#2{%
3932 \endgroup
3933 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3934 \def\mplibtemp@a{#2}%
3935 \ifx\mplib@mplibcode\mplibtemp@a
3936 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]===],"\currentmpinstancename")}%
3937 \end{mplibcode}%
3938 \else
3939 \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3940 \expandafter\ltxdomplibcode
3941 \fi
3942 }
3943 \fi

```

User settings.


```

3944 \def\mplibshowlog#1{\directlua{
3945   local s = string.lower("#1")
3946   if s == "enable" or s == "true" or s == "yes" then
3947     luampLib.showlog = true
3948   else
3949     luampLib.showlog = false
3950   end
3951 }}
3952 \def\mpliblegacybehavior#1{\directlua{
3953   local s = string.lower("#1")
3954   if s == "enable" or s == "true" or s == "yes" then
3955     luampLib.legacyverbatim = true
3956   else
3957     luampLib.legacyverbatim = false
3958   end
3959 }}
3960 \def\mplibverbatim#1{\directlua{
3961   local s = string.lower("#1")
3962   if s == "enable" or s == "true" or s == "yes" then
3963     luampLib.verbatiminput = true
3964   else
3965     luampLib.verbatiminput = false
3966   end
3967 }}
3968 \newtoks\mplibtmptoks

```

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

```

3969 \ifcsname ver@luampLib.sty\endcsname
3970   \protected\def\everymplib{%
3971     \begingroup
3972     \mplibsetupcatcodes
3973     \mplibdoeverymplib
3974   }
3975   \protected\def\everyendmplib{%
3976     \begingroup
3977     \mplibsetupcatcodes
3978     \mplibdoeveryendmplib
3979   }
3980   \newcommand\mplibdoeverymplib[2][{}]{%
3981     \endgroup
3982     \directlua{
3983       luampLib.everymplib["#1"] = [===[\unexpanded{#2}]===]
3984     }%
3985   }
3986   \newcommand\mplibdoeveryendmplib[2][{}]{%
3987     \endgroup
3988     \directlua{
3989       luampLib.everyendmplib["#1"] = [===[\unexpanded{#2}]===]
3990     }%

```

```

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

```

TeX macros for dimen/color

```

4019 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
4020 \def\mpcolor#1#{\domplibcolor{#1}}
4021 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

4022 \def\mplibnumbersystem#1{\directlua{
4023   local t = "#1"
4024   if t == "binary" then t = "decimal" end
4025   luamplib.numbersystem = t
4026 }}

```

Settings for .mp cache files.

```

4027 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4028 \def\mplibdomakenocache#1,{%
4029   \ifx\empty#1\empty
4030     \expandafter\mplibdomakenocache
4031   \else
4032     \ifx\stop#1\else
4033       \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
4034       \expandafter\expandafter\expandafter\mplibdomakenocache

```

```

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

```

More user settings.

```

4050 \def\mplibtexttextlabel#1{\directlua{
4051 local s = string.lower("#1")
4052 if s == "enable" or s == "true" or s == "yes" then
4053 luamplib.texttextlabel = true
4054 else
4055 luamplib.texttextlabel = false
4056 end
4057 }}
4058 \def\mplibcodeinherit#1{\directlua{
4059 local s = string.lower("#1")
4060 if s == "enable" or s == "true" or s == "yes" then
4061 luamplib.codeinherit = true
4062 else
4063 luamplib.codeinherit = false
4064 end
4065 }}
4066 \def\mplibglobaltexttext#1{\directlua{
4067 local s = string.lower("#1")
4068 if s == "enable" or s == "true" or s == "yes" then
4069 luamplib.globaltexttext = true
4070 else
4071 luamplib.globaltexttext = false
4072 end
4073 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

4074 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4075 \def\mplibstarttoPDF#1#2#3#4{%
4076 \prependtomplibbox
4077 \hbox dir TLT\bgroup

```

```

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

```

Text items have a special handler.

```

4108 \def\mplibtexttext#1#2#3#4#5{%
4109 \begingroup
4110 \setbox\mplibscratchbox\hbox
4111 {\font\temp=#1 at #2bp%
4112 \temp
4113 #3}%
4114 \setbox\mplibscratchbox\hbox
4115 {\hskip#4 bp%
4116 \raise#5 bp%
4117 \box\mplibscratchbox}%
4118 \wd\mplibscratchbox0pt%
4119 \ht\mplibscratchbox0pt%
4120 \dp\mplibscratchbox0pt%
4121 \box\mplibscratchbox
4122 \endgroup
4123 }

```

Input luamplib.cfg when it exists.

```

4124 \openin0=luamplib.cfg
4125 \ifeof0 \else
4126   \closein0
4127   \input luamplib.cfg
4128 \fi

```

Code for tagpdf

```

4129 \def\luamplibtagtextboxset#1#2{#2}
4130 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4131 \let\luamplibtagasgroupset\relax
4132 \let\luamplibtagasgroupput\luamplibtagtextboxset
4133 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4134 \ifcsname ver@tagpdf.sty\endcsname \else
4135   \ExplSyntaxOn
4136   \keys_define:nn{luamplib/tagging}
4137   {
4138     ,alt          .code:n = { }
4139     ,actualtext   .code:n = { }
4140     ,artifact     .code:n = { }
4141     ,text         .code:n = { }
4142     ,off          .code:n = { }
4143     ,tag          .code:n = { }
4144     ,adjust-BBox .code:n = { }
4145     ,tagging-setup .code:n = { }
4146     ,instance     .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4147     ,instancename .meta:n = { instance = {#1} }
4148     ,unknown      .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4149   }
4150   \RenewDocumentCommand\mplibcode{0{}}
4151   {
4152     \tl_gclear:N \currentmpinstancename
4153     \keys_set:ne{luamplib/tagging}{#1}
4154     \mplibtmptoks{}\ltxdomplibcode
4155   }
4156   \cs_set_eq:NN \mplibaltext \use_none:n
4157   \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.

```

4158   \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4159   \ExplSyntaxOff
4160   \endinput\fi
4161 \ExplSyntaxOn
4162 \tl_new:N \l__luamplib_tag_envname_tl
4163 \tl_new:N \l__luamplib_tag_alt_tl
4164 \tl_new:N \l__luamplib_tag_alt_dflt_tl

```

```

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

```

Sockets for tex-text boxes.

```

4193 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4194 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4195 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4196 {

```

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.

```

4197 \bool_if:NTF \l__luamplib_tag_usetext_bool
4198 {
4199   \tag_mc_end_push:
4200   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4201   \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.

```

4202   \tag_mc_begin:n{tag=text}
4203   #2
4204   \tag_mc_end:
4205   \tag_struct_end:
4206   \tag_mc_begin_pop:n{ }
4207 }

```

```

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

```

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

```

4250 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4251 {
4252   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usertext_bool
4253   \bool_set_false:N \l__luamplib_tag_usertext_bool

```

```

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

```

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

```

4279 \cs_set_nopar:Npn \luamplibtagasgroupset
4280 {
4281 \bool_set_false:N \l__luamplib_tag_usetext_bool
4282 }
4283 \cs_set_nopar:Npn \luamplibtagasgroupput
4284 {
4285 \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4286 \tag_socket_use:nnn{luamplib/mplibgroup/put}
4287 }

```

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

```

4288 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4289 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4290 {
4291 \cs_if_free:cT {luamplib.mplibgroup.text.#1}
4292 {
4293 \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4294 \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4295 }
4296 \tag_mc_end:
4297 \tag_mc_begin:n{tag=text}
4298 #2

```



```

4299 \tag_mc_end:
4300 \tag_mc_begin:n{artifact}
4301 }
4302 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

A macro for BBox attribute

```

4303 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4304 {
4305   \tl_set:Nc \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4306   \tex_savepos:D
4307   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4308   \tl_set:Nc \l__luamplib_BBox_llx_tl
4309     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4310   \tl_set:Nc \l__luamplib_BBox_lly_tl
4311     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4312   \tl_set:Nc \l__luamplib_BBox_urx_tl
4313     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4314   \tl_set:Nc \l__luamplib_BBox_ury_tl
4315     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4316   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4317   {
4318     \int_zero:N \l_tmpa_int
4319     \tl_map_inline:nn
4320     {
4321       \l__luamplib_BBox_llx_tl
4322       \l__luamplib_BBox_lly_tl
4323       \l__luamplib_BBox_urx_tl
4324       \l__luamplib_BBox_ury_tl
4325     }
4326     {
4327       \int_incr:N \l_tmpa_int
4328       \tl_set:Nc ##1
4329       {
4330         \fp_eval:n
4331         {
4332           ##1
4333           +
4334           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4335         }
4336       }
4337     }
4338   }
4339   \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4340   {
4341     /O /Layout /BBox [
4342       \l__luamplib_BBox_llx_tl\c_space_tl
4343       \l__luamplib_BBox_lly_tl\c_space_tl
4344       \l__luamplib_BBox_urx_tl\c_space_tl
4345       \l__luamplib_BBox_ury_tl

```

```

4346   ]
4347 }
4348 \bool_if:NT \l__tag_graphic_debug_bool
4349 {
4350   \iow_log:e
4351   {
4352     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4353     \l__luamplib_BBox_llx_tl\c_space_tl
4354     \l__luamplib_BBox_lly_tl\c_space_tl
4355     \l__luamplib_BBox_urx_tl\c_space_tl
4356     \l__luamplib_BBox_ury_tl
4357   }
4358   \sys_if_output_pdf:TF
4359   {
4360     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
4361     {
4362       \pdfextension save\relax
4363       \opacity_select:n{0.5} \color_select:n{red}
4364       \pdfextension literal~text
4365       {
4366         \l__luamplib_BBox_llx_tl\c_space_tl
4367         \l__luamplib_BBox_lly_tl\c_space_tl
4368         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4369         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4370         re~f
4371       }
4372       \pdfextension restore\relax
4373     }
4374   }
4375   {
4376     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
4377     {
4378       \special{pdf:bcontent}
4379       \opacity_select:n{0.5} \color_select:n{red}
4380       \special{pdf:code~
4381         1~0~0~1~
4382         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4383         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4384         cm
4385       }
4386       \special{pdf:code~
4387         \l__luamplib_BBox_llx_tl\c_space_tl
4388         \l__luamplib_BBox_lly_tl\c_space_tl
4389         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4390         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4391         re~f
4392       }
4393       \special{pdf:econtent}
4394     }

```

```

4395     }
4396   }
4397 }

```

Sockets for main process

```

4398 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4399 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4400 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4401 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4402 {
4403   \tag_mc_end_push:
4404   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4405   {
4406     \tl_if_empty:eTF{#1}
4407     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4408     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4409     \msg_warning:nnVV{luamplib}{alt-text-missing}
4410     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4411   }
4412   \tag_struct_begin:n
4413   {
4414     tag=\l__luamplib_tag_struct_tl,
4415     alt=\l__luamplib_tag_alt_tl,
4416   }
4417   \tag_mc_begin:n{}
4418 }
4419 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4420 {
4421   \__luamplib_tag_bbox_attribute:n {#1}
4422   #2
4423   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4424   \tag_mc_end:
4425   \tag_struct_end:
4426   \tag_mc_begin_pop:n{}
4427 }
4428 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4429 {
4430   \tag_mc_end_push:
4431   \tag_struct_begin:n
4432   {
4433     tag=Span,
4434     actualtext=\l__luamplib_tag_actual_tl,
4435   }
4436   \tag_mc_begin:n{}
4437 }
4438 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4439 {
4440   #2
4441   \tag_mc_end:

```

```

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

```

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

```

4456 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4457 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4458 {
4459 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4460 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4461 }
4462 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4463 {
4464 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4465 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4466 \prependtomplibbox \mplibnoforcehmode
4467 \mode_if_vertical:T { \noindent \aftergroup\par }
4468 }
4469 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4470 {
4471 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4472 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4473 }
4474 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4475 {
4476 \bool_set_true:N \l__luamplib_tag_usetext_bool
4477 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4478 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4479 \prependtomplibbox \mplibnoforcehmode
4480 \mode_if_vertical:T { \noindent \aftergroup\par }
4481 }
4482 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4483 {
4484 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4485 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4486 }

```

```
4487 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
```

Key-value options

```
4488 \keys_define:nn{luamplib/tagging}
4489 {
4490   ,alt .code:n =
4491   {
4492     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4493     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4494   }
4495   ,actualtext .code:n =
4496   {
4497     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4498     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4499   }
4500   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4501   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4502   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4503   ,tag .code:n =
4504   {
4505     \str_case:nnF {#1}
4506     {
4507       {false} { \keys_set:nn {luamplib/tagging} {off} }
4508       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4509     }
4510     {
4511       \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4512       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4513     }
4514   }
4515   ,adjust-BBox .code:n =
4516   {
4517     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4518     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4519   }
4520   ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4521 }
4522 \keys_define:nn {luamplib/instance}
4523 {
4524   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4525   ,instancename .meta:n = { instance = {#1} }
4526   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4527 }
```

Redefine our macros

```
4528 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4529 {
4530   \prependtomplibbox
4531   \hbox dir~TLT\bgroup
4532   \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
```

```

4533 \xdef\MPllx{#1}\xdef\MPlly{#2}%
4534 \xdef\MPurx{#3}\xdef\MPury{#4}%
4535 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4536 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4537 \parskip0pt
4538 \leftskip0pt
4539 \parindent0pt
4540 \everypar{}%
4541 \setbox\mplibscratchbox\vbox\bgroup
4542 \tag_suspend:n{\mplibstarttoPDF}
4543 \noindent
4544 }
4545 \cs_set_nopar:Npn \mplibstoptoPDF
4546 {
4547 \par
4548 \egroup
4549 \setbox\mplibscratchbox\hbox
4550 {\hskip-\MPllx bp
4551 \raise-\MPlly bp
4552 \box\mplibscratchbox}%
4553 \setbox\mplibscratchbox\vbox to \MPheight
4554 {\vfill
4555 \hsize\MPwidth
4556 \wd\mplibscratchbox0pt
4557 \ht\mplibscratchbox0pt
4558 \dp\mplibscratchbox0pt
4559 \box\mplibscratchbox}%
4560 \wd\mplibscratchbox\MPwidth
4561 \ht\mplibscratchbox\MPheight
4562 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4563 \egroup
4564 }
4565 \RenewDocumentCommand\mplibcode{0{}}
4566 {
4567 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4568 \tl_gclear:N \currentmpinstancename
4569 \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4570 \keys_set:nV {luamplib/instance} \l_tmpa_tl
4571 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4572 \tag_socket_use:n{luamplib/figure/init}
4573 \mplibtmptoks{}\ltxdomplibcode
4574 }
4575 \RenewDocumentCommand\mpfig{s 0{}}
4576 {
4577 \begingroup
4578 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4579 \keys_set_known:ne {luamplib/tagging} {#2}
4580 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4581 \tag_socket_use:n{luamplib/figure/init}

```

```

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

```

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

```

4601 \cs_new_nopar:Npn \mplibalttext #1
4602 {
4603   \tl_set:Ne \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4604 }
4605 \cs_new_nopar:Npn \mplibactualtext #1
4606 {
4607   \tl_set:Ne \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4608 }
4609 \ExplSyntaxOff

```

That's all folks!

3 The GNU GPL License v2

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

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

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

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

Preamble

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

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

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

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

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

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NO WARRANTY

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

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

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

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

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

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

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

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

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

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

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

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

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

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

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

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