

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/10/02 V2.44.1

Abstract

Package to have METAPOST code typeset directly in a document with LuaTeX

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	4
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	\mplibglobaltexttext	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	\mpfig, \endmpfig	8
1.1.15	About cache files	9
1.1.16	About figure box metric	9
1.1.17	luamplib.cfg	9
1.1.18	Tagged PDF	9
1.1.19	Externalize METAPOST figures	11
1.2	METAPOST	13
1.2.1	TeX dimen and color	13
1.2.2	More on TeX color	13
1.2.3	Fill and stroke colors	14

1.2.4	Transparency	14
1.2.5	Fill and stroke transparency	14
1.2.6	Text outline as a text image	15
1.2.7	Glyph outline	15
1.2.8	Drawing a glyph outline	16
1.2.9	Text outline as a path image	16
1.2.10	Unicode-aware length	17
1.2.11	Unicode-aware substring	17
1.2.12	Shading	17
1.2.13	Fading	20
1.2.14	Tiling pattern	21
1.2.15	Form XObject from METAPOST side	24
1.2.16	Form XObject from T _E X side	27
1.2.17	Masking	28
1.2.18	Free-form Gouraud-shaded triangle mesh shading	29
1.2.19	Lattice-form Gouraud-shaded triangle mesh shading	30
1.2.20	Coons patch mesh shading	31
1.2.21	Tensor-product patch mesh shading	33
1.3	Lua	34
1.3.1	runscript	34
1.3.2	Accessing METAPOST variables	35
1.3.3	Running a METAPOST code	35
1.3.4	Registering a tiling pattern	35
1.3.5	Registering a form XObject	36
2	Implementation	36
2.1	Lua module	36
2.2	T _E Xpackage	126
3	The GNU GPL License v2	148

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-mplib.lua` and `luatex-mplib.tex` files from Con $\TeX$ t. They have been adapted to $\LaTeX$ and Plain by Elie Roux and Philipp Gesang and new functionalities have been added by Kim Dohyun. The most notable changes are:

- Possibility to use `btex ... etex` to typeset $\TeX$ code. `texttext <string>` is a more versatile macro equivalent to `TEX <string>` from `TEX.mp`. `TEX` is also allowed and is a synonym of `texttext`. The argument of `mplib`'s primitive `maketext` will also be processed by the same routine.
- Possibility to use `verbatimtex ... etex` to run a $\TeX$ code. `VerbatimTeX <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}
```



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.

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>}`.

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

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

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

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

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

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

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

where `<string>` should be "" (empty), "isolated", "knockout", or "isolated, knockout". Beware that currently many of the PDF rendering applications, except Adobe Acrobat and T_EXworks among others, cannot properly render the isolated or knockout effect.

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

1.1.4 Choosing a number system

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

1.1.5 Showing METAPOST log

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

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

1.1.6 verbatimtex Legacy behavior

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

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

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

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

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



diameter: 22.62764bp.

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

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

ABC **DEF GHI**

1.1.7 $\TeX$ -text labels

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

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

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

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

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

1.1.8 Inherit METAPOST code

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

1.1.9 \mplibglobaltexttext{enable|disable}

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

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



1.1.10 Separate METAPOST instances

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

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

The behavior is as follows.

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

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

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

1.1.11 Verbatim input

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

1.1.12 $\TeX$ dimen

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

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



1.1.13 $\TeX$ color

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

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

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

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



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

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

1.1.14 `\mpfig ... \endmpfig`

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

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

and the starred version is roughly the same as follows:

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

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

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



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

1.1.15 About cache files

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

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

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

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

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

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

1.1.16 About figure box metric

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

1.1.17 `luamplib.cfg`

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

1.1.18 Tagged PDF

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1.1.19 Externalize METAPOST figures

Though `mplib` library is already quite fast, generating external image files from `METAPOST` figures and including them as `XObjects` into the PDF could significantly improve compilation speed. Inspired by `PGF`'s 'external' library, `luamplib` also provides similar functionality: the *externalize* feature, which requires `--shell-escape` command-line option.

You can invoke this feature by the command `\mplibexternalize`. Though the command should be used only once, its location is not limited to the preamble.

Other macros for 'externalize' feature:

\mplibexternalizename $\langle$ *prefix* $\rangle$ The filename of external images starts with $\langle$ jobname $\rangle$ -mplib-figure by default, which you can change using this command. `\mplibexternalizename{figures/metapost-figure}`, for instance, will generate `metapost-figures.pdf` or (in case you have modified `METAPOST` code after the initial run) `metapost-figure-*.pdf` files in the `figures` subdirectory under the directory where output files will be written. The subdirectory should be existent in advance.

N.B. You should put this command *before* `\mplibexternalize`.

N.B. If something goes wrong with ‘externalize’ feature, delete the files whose name starts with $\langle prefix \rangle$ (especially the .lua file). At next run of $\text{\LaTeX}$, the world of externalization will start anew from the very beginning. Again, it would be a good idea to delete the files just mentioned when you have modified lots of code chunks after the initial run of externalization, because it will improve the compilation speed and make the size of PDF file much smaller.

`\mplibexternalizesuspend` The operation of ‘externalize’ functionality (including the increment of a figure number) will be halted. This command does not respect $\text{\TeX}$ ’s group scope.

`\mplibexternalizeresume` This command, which also disregards $\text{\TeX}$ ’s group scope, resumes the ‘externalize’ feature.

Options to `\mplibcode` environment, and to the `\mpfig` and `\usemplibgroup` commands:

`externalize= $\langle boolean \rangle$` Unless you redefine a low-level command, option `externalize` (short-cut of `externalize=true`) does nothing. However, when `externalize=false` is given, external image(s) for the specific code chunk will not be generated even if ‘externalize’ feature is enabled.

`externalizedependson= $\{\langle number \rangle, \dots\}$` If you modify a `METAPOST` code chunk which is dependent on previous code chunks (this occurs when `\mplibcodeinherit` or `\mplib` instance name is used), `luamplib`, not knowing the dependency, feeds only the changed code chunk to the `\mplib` library. This will raise a `METAPOST` error. To prevent the error, you have to specify $\langle number \rangle$ (s) of the previous code chunk(s) by using this option, for instance `externalizedependson={2,3}`.

The $\langle number \rangle$ is set to zero by the `\mplibexternalize` command and increases by one each time `luamplib` meets `\mplibcode`, `\mpfig`, or `\usemplibgroup`. If the given $\langle number \rangle$ is negative, `luamplib` searches for the code chunk by tracing backwards: -1 refers to the immediately preceding chunk.

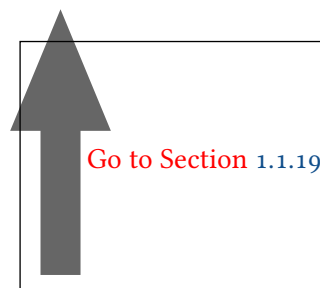
`externalizemargin= $\langle dimen \rangle$` To address the cutting-off problem (see the limitations below), we provide this option which will enlarge the BBox of external image(s) by the amount of $\langle dimen \rangle$. As it affects only PDF BBox values, the metric of $\text{\TeX}$ boxes remains the same.

Known limitations

- The ‘externalize’ feature currently supports only $\text{\LaTeX}$ format in PDF mode.
- The part of a figure that protrudes out of the figure’s bounding box will be cut off, and so invisible. As said, `externalizemargin` option is provided to address this problem.
- Hyperlinks in `btex` ... `etex` (and its derivative) commands will be broken.

- As the structure of resulting PDF is quite complicated, some PDF viewers cannot render properly the embedded images that contain special effects we provide (to be precise, fading or masking).

```
\mplibexternalize
\mpfig [externalizemargin=12bp]
  interim bboxmargin := 0 ;
  interim linejoin := mitered ;
  interim linecap := butt ;
  draw image(drawarrow (down--up) scaled 8
    withpen pencircle scaled 3)
    scaled 5 withcolor .4 white ;
  draw TEX "Go to Section \ref{sec:externalize}."
    shifted 10 right withcolor red ;
  draw bbox currentpicture ;
\endmpfig
\mplibexternalizesuspend
```



Therefore we recommend that users use the ‘externalize’ functionality only privately, such as for compiling while editing the document, not for final PDF to be distributed publicly.

1.2 METAPOST

1.2.1 T_EX dimen and color

`mplibdimen` *<string>* and `mplibcolor` *<string>* 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` *<string>* 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 T_EX color. Therefore the example shown above would raise a METAPOST error: `cmykcolor col;` should have been declared. By contrast, `mplibrgbtexcolor` *<string>* always returns rgb-model expressions.

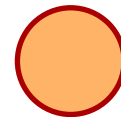
N.B. Spot colors are forced to cmyk or rgb model, so these operators are not recommended for spot colors.

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

1.2.3 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 (<fill color expr>, <stroke color expr>)`. When the argument is in string type, it is regarded as the color expression of T_EX 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) ;
```

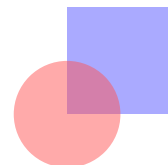


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 Transparency

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

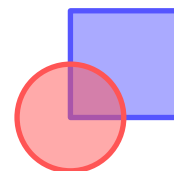
```
\mpfig
  fill unitsquare scaled 40
    withcolor 1/3[blue,white]
    withtransparency (1, 0.5)      % or ("normal", 0.5)
  ;
  fill fullcircle scaled 40
    withcolor 1/3[red,white]
    withtransparency (1, 0.5)
  ;
\endmpfig
```



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

`mplibgraphictext` *<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 mplibgraphictext "$\sqrt{2+x}$"
  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 `mplibgraphictext` 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, `mplibgraphictext` 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.

1.2.7 Glyph outline

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

```

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

```

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.

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

N.B. Regrettably we have some bug in processing not a few glyphs in `cmr10.pfb` and its family (or maybe other) Type1 fonts.¹² If that happens, consider using glyph operator instead of `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` *<picture>* command fills the paths according to the nonzero winding number rule. As a result, for instance, the area surrounded by inner path of “O” will remain transparent.

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

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

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

An example:

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



1.2.9 Text outline as a path image

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

¹²The bug seems to be fixed in `font-cff.lmt` contained in ConT_EXt mkx_l, but current `luaotfload` is based on `font-cff.lua` from ConT_EXt mkiv. As you see, `mplibglyph` operator requires `luaotfload` package loaded, which however is done automatically by L^AT_EX format.

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

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` *<string>* returns the number of unicode characters in the string. This is a unicode-aware version equivalent to the METAPOST primitive `length`, but accepts only a string-type argument. For instance, `mpliblength "abçdéf"` returns 6, not 8.

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

1.2.11 Unicode-aware substring

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

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

1.2.12 Shading

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

- *Textual pictures* as well as paths can have shading effect. The term *textual picture* here means a picture generated by `btex ... etex`, `texttext`, `TEX`, `maketext`, `mplibgraphicstext` (see below § 1.2.6), or `infont` operator, though technically only the last one is a true textual

picture. Note that the picture, including transparency group, in which the objects are painted *without* color can also be regarded as a textual picture.¹⁴

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



- When shading a picture generated by ‘infont’ operator or that has multiple components, the effect of `withshadingvector` and that of `withshadingdirection` will be the same, as `luamplib` considers only the bounding box of the picture.
- A few more optional macros are available in addition to the *metafun*’s: `withshadingpoints`, `withshadingcenters`, `withshadingextend`, `withshadingmatrix`, and `withshadingstroke`.
- More shading methods are available: “triangle”, “lattice”, “coons”, and “tensor”. See below § 1.2.18, § 1.2.19, § 1.2.20, and § 1.2.21 for these methods.

The syntax is `<path> | <textual picture> withshadingmethod <string>`, where the latter shall be “linear”, “circular”, “triangle”, “lattice”, “coons”, or “tensor”. The balance of this subsection is to explain additional optional macros.

Above all, there are two ways in specifying the shading coordinates, of which you can choose the more convenient one. First, the way that mimicks the *metafun*’s:

`withshadingvector` `<pair>` Starting and ending points (as time value) on the path.

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

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

`withshadingcenter` `<pair>` Value to specify the starting center. For instance, `(0,0)` means that the center of starting circle is center `p`; `(1,1)` means `urcorner p`; `(-1,-1)` means `llcorner p`.

`withshadingradius` `<pair>` Radii of starting and ending circles. This is no-op in linear mode. Default value: `(0, abs(center p - urcorner p))`

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

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

withshadingtransform $\langle \textit{string} \rangle$ where $\langle \textit{string} \rangle$ 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 ($\langle \textit{pair} \rangle$, $\langle \textit{pair} \rangle$) 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 ($\langle \textit{pair} \rangle$, $\langle \textit{pair} \rangle$) 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.

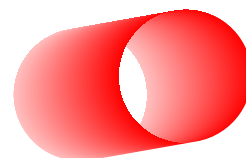
withshadingfraction $\langle \textit{numeric} \rangle$ Fractional number of each shading step, and so only meaningful within withshadingstep.

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

withshadingdomain $\langle \textit{pair} \rangle$ 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 ($\langle \textit{boolean} \rangle$, $\langle \textit{boolean} \rangle$) 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
```

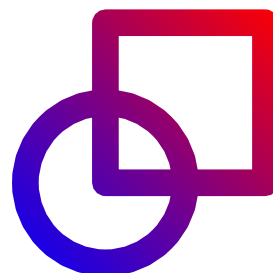


withshadingmatrix $\langle \text{string} \rangle$ METAPOST code for transformation of shading, such as "xscaled 1.2 yscaled 0.8"; or six numerics separated by spaces, such as "1.2 0 0 0.8 0 0" (xx, yx, xy, yy, x, and y-part respectively).

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

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

withfadefraction <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 (BBox) 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.

withadjustbbox <numeric> More convenient way to adjust BBox value than withfadebbox. The lower left corner of the BBox will move leftward and downward by the amount specified, and the upper right corner will move rightward and upward by the same amount.

An example:

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



1.2.14 Tiling pattern

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

An example:

```
\mppattern{mypatt}           % or \begin{mppattern}{mypatt}
[                             % options: see below
  xstep = 10,
  ystep = 7,
  matrix = "rotated 45",      % or "0.7 0.7 -0.7 0.7" or {0.7, 0.7, -0.7, 0.7}
]
```

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.

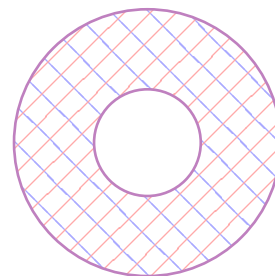
Table 1: options for \mppattern

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

```
\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
\endmpattern          % or \end{mpattern}
```

```
\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.

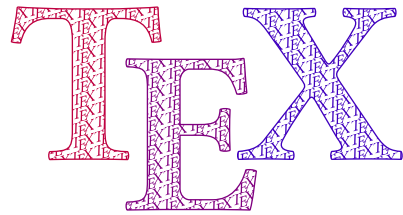
Option `colored=false` (or `coloured=false`) will generate an uncolored pattern cell which shall have no color at all (i.e. `withoutcolor` command is needed for METAPOST code).¹⁶ Uncolored

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

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:

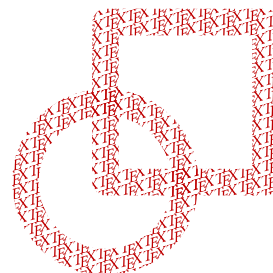
withpatternstroke *<string>* where the string shall be "filldraw", "both", "draw", "yes", or "no":
 "filldraw" or "both" will tile both the fill part and the stroke part; "draw" or "yes" will
 tile the stroke part only; all other cases will tile the fill part only, which is the default
 behavior. An 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 Form XObject from METAPOST side

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

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

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

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

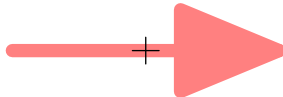
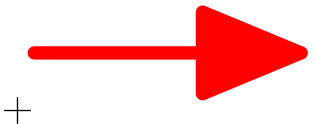
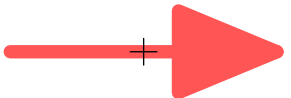
\usemplibgroup{ $\langle name \rangle$ } is a $\text{\TeX}$ command to reuse an XObject of the name once used. Note that the position of the XObject will be origin-based: in other words, lower left corner of the bounding box will be shifted to the origin.

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

withgroupbbox (*<pair>*, *<pair>*) sets the bounding box (BBox) of the XObject, default value being (lrcorner p, urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence ‘withgroupbbox (bot lft llcorner p, top rt urcorner p)’, supposing that the pen was selected by the pickup command. However, this option is not needed in most cases as **luamplib** v2.42.9 or after tries to detect pen width information.

withadjustbbox (*<numeric>*) More convenient way to adjust BBox value than withgroupbbox. The lower left corner of the BBox will move leftward and downward by the amount specified, and the upper right corner will move rightward and upward by the same amount.

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 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

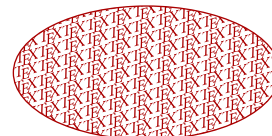
T_EX macros `\mplibgroup{<name>}` ... `\endmplibgroup` are described here in this subsection, as they are deeply related to the `asgroup` operator described just above at § 1.2.15. With these, users can define a transparency group or an ordinary *form XObject* from T_EX side. The syntax is similar to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

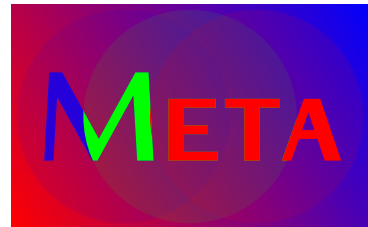
²⁰Color expressions in rgb or cmyk model are also allowed, in accordance with the option colorspace="/DeviceRGB"

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.

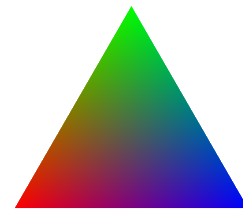
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates).

The third argument is the color for the new vertex.

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

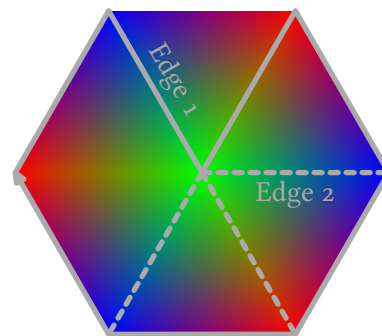
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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.

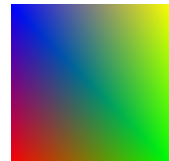
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

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.

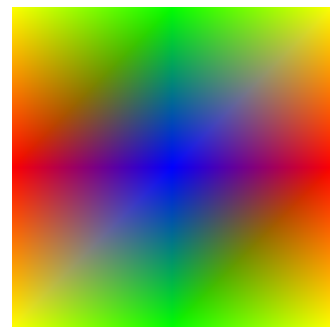
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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.

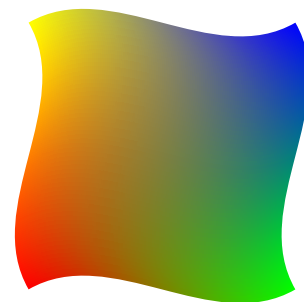
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

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.

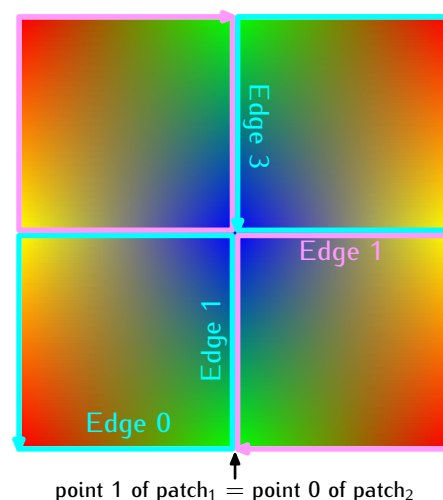
Remaining arguments are two color expressions for the new vertices.

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

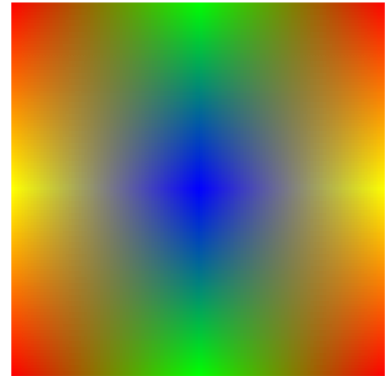
```

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.

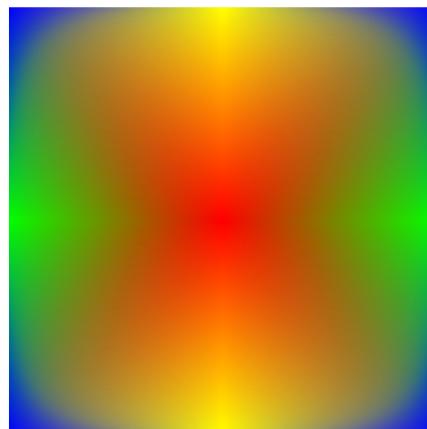
The only difference to withcoonspatchinit macro of Coons patch shading is the second argument inserted for specifying four inner control points. It shall be a path that has four points, or a string composed of eight numerics separated by spaces (x and y coordinates of each point). If the first argument 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 some PDF viewers including Mac OS Preview do not render properly the figure above.

1.3 Lua

1.3.1 runscript ...

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

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

1.3.2 Accessing METAPOST variables

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

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

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

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

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

3.0

1.3.3 Running a METAPOST code

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

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

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

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

1.3.4 Registering a tiling pattern

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

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

Table 3: elements in luamplib table (partial)

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

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 *<pattern name>* before preparing the box, if shading pattern (i.e. shading on picture) is used in the pattern cell.

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

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version    = "2.44.1",
5   date      = "2026/10/02",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8

```

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

```

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

```

Use our own function for warn/info/err.

```

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

```

```

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

```

Provide a few “shortcuts” expected by the code.

```

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

```

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

```

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

```

92

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

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

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

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

```

```

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

```

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

```

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

```

```

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

```

Function `luamplib.finder`

```

210 local exeidx = 0 -- index of the executable
211 while arg[exeidx-1] do
212     exeidx = exeidx - 1
213 end
214 local currenttime = os.time()
215 do
216     local luamplibtime = lfsattributes(kpse.find_file"luamplib.lua", "modification")

```

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

```

217 local function replaceformatmp(file,newfile,ofmodify)
218     local fh = ioopen(file,"r")
219     if not fh then return file end
220     local data = fh:read("*all"); fh:close()

```

```

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

```

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

```

250 local count
251 data, count = scan_mpcode(data, replace_texblock)
252 if count == 0 then
253   needtoreplace[name] = true
254   fh = ioopen(newfile,"w");
255   if fh then
256     fh:close()
257     lfstouch(newfile,currenttime,ofmodify)
258   end
259   return file
260 end
261 fh = ioopen(newfile,"w")
262 if not fh then return file end
263 fh:write(data); fh:close()
264 lfstouch(newfile,currenttime,ofmodify)
265 return newfile
266 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.

```

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

For the main function: process

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

```

296 local currentformat = "plain"
297 function luamplib.setformat (name)
298   currentformat = name
299 end

```

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

```

300 luamplib.codeinherit = false
301 local mplibinstances = {}
302 luamplib.instances = mplibinstances
303 local has_instancename = false
304
305 local process
306 do
307   local function reporterror (result, prevlog)
308     if not result then
309       err("no result object returned")

```

```

310     else
311         local t, e, l = result.term, result.error, result.log
log has more information than term, so log first (2021/08/02)
312         local log = l or t or "no-term"
313         log = log:gsub("(Please type a command or say `end`)", ""):gsub("\n+", "\n")
314         if result.status > 0 then
315             local first = log:match"(.-\n! .-)\n! "
316             if first then
317                 termorlog("term", first)
318                 termorlog("log", log, "Warning")
319             else
320                 warn(log)
321             end
322             if result.status > 1 then
323                 err(e or "see above messages")
324             end
325         elseif prevlog then
326             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.

```

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

```

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

```

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

```

343     make_text   = luamplib.maketext,
344     run_script  = luamplib.runscript,
345     math_mode   = luamplib.numbersystem,
346     job_name    = tex.jobname,
347     random_seed = math.random(4095),
348     utf8_mode   = true,
349     extensions  = 1,

```

```
350 }
```

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

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

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

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

```

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

```

dvipdfmx is supported, though nobody seems to use it.

```

410 local pdfmode = tex.outputmode > 0
411

```

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

```

412 local catlatex = luatexbase.registernumber("catcodetable@latex")
413 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.

```

414 local function run_tex_code (str, cat)
415     texruntoks(function() texsprint(cat or catlatex, str) end)
416 end

```

For conversion of sp to bp.

```

417 local factor = 65536*(7227/7200)
418

```

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

```

419 local texboxes = { globalid = 0, localid = 4096 }
420 local process_tex_text
421 do
422     local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
423         xscaled %f yscaled %f shifted (0,-%f) \z
424         withprescript "mplibtexboxid=%i:%f:%f")'
425     function process_tex_text (str, maketext)
426         if str then
427             if not maketext then str = str:gsub("\r.-$", "") end
428             local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
429                 and "\\global" or ""
430             local tex_box_id
431             if global == "" then
432                 tex_box_id = texboxes.localid + 1

```

```

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

```

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.

```

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

```

A common function for color functions

```

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

```

```

478 return false
479 end
480 local function colorsplit (res)
481   local t, tt = { }, res:explode()
482   local be = tt[1]:find"^%d" and 1 or 2
483   for i=be, #tt do
484     if not tonumber(tt[i]) then break end
485     t[#t+1] = tt[i]
486   end
487   return t
488 end
489 do
490   local colfmt = ccexplat and "l3color" or "xcolor"
491   local mplibcolorfmt = {

```

Fix a bug reported at #207, which was introduced recently when `\edef` was changed to `\expandafter`. We now use `\expanded` instead.

```

492   xcolor = [[{\setbox0\hbox{{\color%s\global\mplibtmptoks\expanded{{\current@color}}}}}],
493   l3color = [[\color_export:nnN{s}{raw}\l_tmpa_tl\mplibtmptoks\expandafter{\l_tmpa_tl}]]
494 }
495 function process_color (str)
496   if str then
497     if not str:find("%b{") then
498       str = format("%s",str)
499     end
500     local myfmt = mplibcolorfmt[colfmt]
501     if colfmt == "l3color" and is_defined"color" then
502       if str:find("%b[") or is_xcolor(str:match"{{(.+)}}") then
503         myfmt = mplibcolorfmt.xcolor
504       end
505     end
506     run_tex_code(myfmt:format(str), ccexplat or catat11)
507     local t = texgettoks"mplibtmptoks"
508     if not pdfmode then
509       if t:find"^hsb" then
510         local spec = t:gsub("^hsb ", ""):gsub(" ", ",")
511         run_tex_code{"\\convertcolorspec{hsb}{", spec, "}{rgb}\\mplibtmpa"}
512         t = format("rgb %s", get_macro"mplibtmpa":gsub(",", " "))
513       elseif not t:find"%d" then -- named color
514         if not is_defined"extractcolorspecs" then err"needs xcolor package loaded" end
515         run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
516         t = format("%s %s", get_macro"mplibtmpa", get_macro"mplibtmpb":gsub(",", " "))
517       end
518       local name, value = t:match"^(%a+) (.+)"
519       if name and value then
520         local op = name == "rgb" and "rg" or name == "cmyk" and "k" or name == "gray" and "g"
521         if not op then err"unknown color model" end
522         t = ("%s %s %s %s"):format(value, op, value, op:upper())
523       end

```

```

524 elseif is_defined"ver@colorspace.sty" and t:find"^/&" then
525   local a,b,c,d = t:match"^(.- cs) (.- CS) (.- scn?) (.- SCN?)"$
526   if a and b and c and d then
527     t = tableconcat({ a, c, b, d }, " ")
528   end
529 end
530 return format('1 withprescript "mpliboverridecolor=%s"', t)
531 end
532 return ""
533 end
534 function process_mplibcolor(str)
535   local res = process_color(str)
536   if res:find" cs " then return res end
537   res = res:match'"mpliboverridecolor=(.)"'
538   local t = colorsplit(res)
539   if #t == 0 then
540     err("Color processing failed '%s'. It could be a luamplib bug. Please report.", res)
541   end
542   return format("(%s)", tableconcat(t, ","))
543 end
544 end
545
546   for \mpdim or mplibdimen
547 local function process_dimen (str)
548   if str then
549     str = str:gsub("{(.+)}", "%1")
550     run_tex_code(format([[ \mplib\mplib\mplib\expandafter{\the\dimexpr %s\relax}]], str))
551     return format("begin group %s end group", texgettoks"mplib\mplib\mplib")
552   end
553   return ""
554 end
555

```

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

```

555 local function process_verbatimtex_text (str)
556   if str then
557     run_tex_code(str)
558   end
559   return ""
560 end
561

```

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.

```

562 local tex_code_pre_mplib = {}
563 luamplib.figid = 1
564 luamplib.in_the_fig = false
565 local function process_verbatimtex_prefig (str)
566   if str then

```

```

567 tex_code_pre_mplib[luamplib.figid] = str
568 end
569 return ""
570 end
571 local function process_verbatimtex_infig (str)
572   if str then
573     return format('special "postmplibverbtex=%s";', str)
574   end
575   return ""
576 end
577

```

For *metafun* format. see issue #79.

```

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

```

metafun 2021-03-09 changes crashes luamplib.

```

583 catcodes = catcodes or {}
584 local catcodes = catcodes
585 catcodes.numbers = catcodes.numbers or {}
586 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
587 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
588 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
589 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
590 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
591 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
592 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
593

```

Now luamplib.runscript

```

594 do
595   local runscript_funcs = {
596     luamplibtext    = process_tex_text,
597     luamplibcolor   = process_mplibcolor,
598     luamplibdimen   = process_dimen,
599     luamplibprefig  = process_verbatimtex_prefig,
600     luamplibinfig   = process_verbatimtex_infig,
601     luamplibverbtex = process_verbatimtex_text,
602   }

```

A function from ConT_EXt general.

```

603 local function mpprint(buffer,...)
604   for i=1,select("#",...) do
605     local value = select(i,...)
606     if value ~= nil then
607       local t = type(value)
608       if t == "number" then
609         buffer[#buffer+1] = format("%.16f",value)
610       elseif t == "string" then

```

```

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

```

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

```

650 local function protecttexcontents (str)
651     return str:gsub("\\%", "\\0Percent\0")
652         :gsub("%%.-\n", "")
653         :gsub("%%.-$", "")
654         :gsub("\\0Percent\0", "\\%")
655         :gsub("%r.-$", "")
656         :gsub("%s+", " ")

```

```

657 end
658 function luamplib.maketext (str, what)
659   if str and str ~= "" then
660     str = protecttexcontents(str)
661     if what == 1 then
662       if not str:find("\\documentclass"..name_e) and
663         not str:find("\\begin%s*{document}") and
664         not str:find("\\documentstyle"..name_e) and
665         not str:find("\\usepackage"..name_e) then
666         if luamplib.legacyverbatim then
667           if luamplib.in_the_fig then
668             return process_verbatimtex_infig(str)
669           else
670             return process_verbatimtex_prefig(str)
671           end
672         else
673           return process_verbatimtex_text(str)
674         end
675       end
676     else
677       return process_tex_text(str, true) -- bool is for 'char13'
678     end
679   end
680   return ""
681 end
682 end
683

```

luamplib's METAPOST color operators

```

684 local function get_spc_name_obj (spot)
685   if is_defined"spc@csall" then
686     for v in get_macro"spc@csall":gmatch"{{(.-)}}" do
687       local n, r = get_macro("spc@ir@".v):match"^(.-) (%d+ 0 R)"
688       if spot == n then
689         return v, r
690       end
691     end
692   else
693     err"only l3color or colorspace is supported for spot color"
694   end
695 end
696 do
697   local function cmyk2rgb (t)
698     return {
699       1 - math.min(1, t[1]+t[4]),
700       1 - math.min(1, t[2]+t[4]),
701       1 - math.min(1, t[3]+t[4]),
702     }
703   end
704   luamplib.gettexcolor = function (str, rgb)

```

```

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

```

An example of spot color shading:

```

\DocumentMetadata{ }
\documentclass{article}
\usepackage{luamplib}
\ExplSyntaxOn
\color_model_new:nnn { pantone3005 }
{ Separation }
{
  name = PANTONE~3005~U ,
  alternative-model = cmyk ,
  alternative-values = {1, 0.56, 0, 0}
}
\color_set:nnn{spotA}{pantone3005}{1}
\color_set:nnn{spotB}{pantone3005}{0.6}
\color_model_new:nnn { pantone1215 }
{ Separation }
{
  name = PANTONE~1215~U ,
  alternative-model = cmyk ,

```

```

        alternative-values = {0, 0.15, 0.51, 0}
    }
    \color_set:nnn{spotC}{pantone1215}{1}
\ExplSyntaxOff
\begin{document}
\begin{mplibcode}
beginfig(1)
    fill unitsquare xscaled \mpdim\textwidth yscaled 1cm
        withshadingmethod "linear"
        withshadingvector (0,1)
        withshadingstep (
            withshadingfraction .5
            withshadingcolors ("spotB","spotC")
        )
        withshadingstep (
            withshadingfraction 1
            withshadingcolors ("spotC","magenta!50")
        )
    ;
endfig;
\end{mplibcode}
\end{document}

```

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

```

\DocumentMetadata{ }
\documentclass{article}
\usepackage{luamplib}
\ExplSyntaxOn
\color_model_new:nnn { pantone1215 }
{ Separation }
{
    name = PANTONE~1215~U ,
    alternative-model = cmyk ,
    alternative-values = {0, 0.15, 0.51, 0}
}
\color_model_new:nnn { pantone+black }
{ DeviceN }
{ names = {pantone1215,black} }
\color_set:nnn{purepantone}{pantone+black}{1,0}
\color_set:nnn{pureblack}{pantone+black}{0,1}
\ExplSyntaxOff
\begin{document}
\mpfig
    fill unitsquare xscaled \mpdim{\textwidth} yscaled 30
        withshadingmethod "linear"
        withshadingcolors ("purepantone","pureblack")
    ;
\endmpfig

```

```

\end{document}

735 local name, value, obj
736 if not is_xcolor(str) then
737   run_tex_code({ "\\color_export:nnN{", str, "{backend}\\mplib@tempa" }, ccexplat)
738   name, value = get_macro'mplib@tempa':match'{{(.-)}}{(.-)}'
739   local t = res:explode()
740   local refname = t[1]:sub(2,-1)
741   if pdfmode then
742     obj = format("%s 0 R", ltx.pdf.object_id(refname))
743   else
744     run_tex_code({ "\\mplibtmp toks\\expanded{{\\pdf_object_ref:n{", refname, "}}}" }, ccexplat)
745     obj = texgettoks"mplibtmp toks"
746   end
747 else -- colorspace: mixing is allowed only in preamble
748   name, value = res:match"^(.-) cs (.-) sc"
749   name, obj = get_spc_name_obj(name)
750 end
751 return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
752 end
753 return format('(%s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),","), str)
754 end
755

```

luamplib.fillandstrokecolor

```

756 do
757 local function graphictextcolor (col, filldraw)
758   if col:find"^[%d%.:]+$" then
759     col = col:explode"."
760     for i=1,#col do
761       col[i] = format("%.3f", col[i])
762     end
763     local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
764     col[#col+1] = filldraw == "fill" and op or op:upper()
765     return tableconcat(col," ")
766   end
767   col = process_color(col):match'"mpliboverridecolor=(.+)"'
768   local t = col:explode()
769   local b = filldraw == "fill" and 1 or #t/2+1
770   local e = b == 1 and #t/2 or #t
771   return tableconcat(t," ", b, e)
772 end
773 function luamplib.fillandstrokecolor (fill, stroke)
774   fill = graphictextcolor(fill, "fill")
775   stroke = graphictextcolor(stroke, "stroke")
776   return format('withprescript "mpliboverridecolor=%s %s"', fill, stroke)
777 end
778 end
779

```

Remove trailing zeros for smaller PDF

```
780 local decimals = "%. %d+"
781 local function rmzeros(str) return str:gsub("%.?0+$", "") end
782
```

common function for mplibgraphicstext and mpliboutlinetext

```
783 local function getrulemetric (box, curr, bp)
784   local running = -1073741824
785   local wd,ht,dp = curr.width, curr.height, curr.depth
786   wd = wd == running and box.width or wd
787   ht = ht == running and box.height or ht
788   dp = dp == running and box.depth or dp
789   if bp then
790     return wd/factor, ht/factor, dp/factor
791   end
792   return wd, ht, dp
793 end
794
```

luamplib's mplibgraphicstext operator

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

```

825     return width
826 end
827 local function getrulewhatsit (line, wd, ht, dp)
828     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
829     line = line == 0 and "" or ("%f w"):format(line)
830     local pl
831     local fmt = "q %s %f %f %f %f re B Q"
832     if pdfmode then
833         pl = node.new("whatsit", "pdf_literal")
834         pl.mode = 0
835     else
836         fmt = "pdf:content " .. fmt
837         pl = node.new("whatsit", "special")
838     end
839     pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
840     local ss = node.new"glue"
841     node.setglue(ss, 0, 65536, 65536, 2, 2)
842     pl.next = ss
843     return pl
844 end

```

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

```

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

```

```

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

```

```

921     end
922     curr.font = i
923   end
924   ::skip_type1::
925   curr = node.getnext(curr)
926 end
927 return head
928 end
929 luamplib.graphicstext = function (text, fakebold, fc, dc)
930   local fmt = process_tex_text(text):sub(1,-2)
931   local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
932   emboldenfonts.width = nil
933   local box = texgetbox(id)
934   box.head = embolden(box, box.head, fakebold)
935   local colors = luamplib.fillandstrokecolor(fc, dc)
936   return format('%s %s)', fmt, colors)
937 end
938 end
939

```

luamplib's mplibglyph operator

```

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

```

```

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

```

```

1017     return format('image(%s)', tableconcat(r)), towarn
1018 end
1019 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
1020 function luamplib.glyph (f, c)
1021     local filename, subfont, instance, kind, shapedata
1022     local fid = tonumber(f) or font.id(f)
1023     if fid > 0 then
1024         local fontdata = font.getfont(fid) or font.getcopy(fid)
1025         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
1026         instance = fontdata.specification and fontdata.specification.instance
1027             or fontdata.shared and fontdata.shared.features.axis
1028         filename = filename and filename:gsub("^harfloaded:", "")
1029     else
1030         local name
1031         f = f:match"^%s*(.)%s*$"
1032         name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
1033         if not name then
1034             name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
1035         end
1036         if not name then
1037             name, subfont = f:match"(.+)%((%d+)%)$" -- Times.ttc(2)
1038         end
1039         name = name or f
1040         subfont = (subfont or 0)+1
1041         instance = instance and instance:lower()
1042         for _,ftype in ipairs{"opentype", "truetype"} do
1043             filename = kpse.find_file(name, ftype.." fonts")
1044             if filename then
1045                 kind = ftype; break
1046             end
1047         end
1048     end
1049     if kind ~= "opentype" and kind ~= "truetype" then
1050         f = fid and fid > 0 and tex.fontname(fid) or f
1051         if kpse.find_file(f, "tfm") then
1052             return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
1053         else
1054             filename = kpse.find_file(f, "type1 fonts")
1055             if filename then
1056                 kind = "type1" -- there's bug in processing cmr family
1057             else
1058                 return mperr"font not found"
1059             end
1060         end
1061     end
1062     local time = lfsattributes(filename,"modification")

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

```

```

1063     local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1064     local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
1065     local newname = format("%s/%s.lua", cachedir or outputdir(), h)
1066     local newtime = lfsattributes(newname,"modification") or 0
1067     if time == newtime then
1068         shapedata = require(newname)
1069     end
1070     if not shapedata then
1071         if fonts then
1072             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1073             shapedata = handler.readers.loadshapes(filename,subfont,instance)
1074         end
1075         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1076         table.toFile(newname, shapedata, "return")
1077         lfstouch(newname, time, time)
1078     end
1079     local gid = tonumber(c)
1080     if not gid then
1081         local uni = utf8.codepoint(c)
1082         for i,v in pairs(shapedata.glyphs) do
1083             if c == v.name or uni == v.unicode then
1084                 gid = i; break
1085             end
1086         end
1087     end
1088     if not gid then return mperr"cannot get GID (glyph id)" end
1089     local fac = 1000 / (shapedata.units or 1000)
1090     local t = shapedata.glyphs[gid]; t = t and t.segments
1091     if not t then return "image()" end
1092     for i,v in ipairs(t) do
1093         if type(v) == "table" then
1094             for ii,vv in ipairs(v) do
1095                 if type(vv) == "number" then
1096                     t[i][ii] = format("%.0f", vv * fac)
1097                 end
1098             end
1099         end
1100     end
1101     local result, towarn = glyphimage(t, shapedata.format or kind)
1102     if towarn then
1103         warn("mplibglyph %s not working properly. Use glyph instead", f)
1104     end
1105     return result
1106 end
1107 end
1108

```

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

```

1109 do

```

```

1110 local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1111   unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1112 local outline_horz, outline_vert
1113 function outline_vert (res, box, curr, xshift, yshift)
1114   local b2u = box.dir == "LTL"
1115   local dy = (b2u and -box.depth or box.height)/factor
1116   local ody = dy
1117   while curr do
1118     if curr.id == node.id"rule" then
1119       local wd, ht, dp = getrulemetric(box, curr, true)
1120       local hd = ht + dp
1121       if hd ~= 0 then
1122         dy = dy + (b2u and dp or -ht)
1123         if wd ~= 0 and curr.subtype == 0 then
1124           res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1125         end
1126         dy = dy + (b2u and ht or -dp)
1127       end
1128     elseif curr.id == node.id"glue" then
1129       local vwidth = node.effective_glue(curr,box)/factor
1130       if curr.leader then
1131         local curr, kind = curr.leader, curr.subtype
1132         if curr.id == node.id"rule" then
1133           local wd = getrulemetric(box, curr, true)
1134           if wd ~= 0 then
1135             local hd = vwidth
1136             local dy = dy + (b2u and 0 or -hd)
1137             if hd ~= 0 and curr.subtype == 0 then
1138               res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1139             end
1140           end
1141         elseif curr.head then
1142           local hd = (curr.height + curr.depth)/factor
1143           if hd <= vwidth then
1144             local dy, n, iy = dy, 0, 0
1145             if kind == 100 or kind == 103 then -- todo: gleaders
1146               local ady = abs(ody - dy)
1147               local ndy = math.ceil(ady / hd) * hd
1148               local diff = ndy - ady
1149               n = math.floor((vwidth-diff) / hd)
1150               dy = dy + (b2u and diff or -diff)
1151             else
1152               n = math.floor(vwidth / hd)
1153               if kind == 101 then
1154                 local side = vwidth % hd / 2
1155                 dy = dy + (b2u and side or -side)
1156               elseif kind == 102 then
1157                 iy = vwidth % hd / (n+1)
1158                 dy = dy + (b2u and iy or -iy)

```

```

1159         end
1160     end
1161     dy = dy + (b2u and curr.depth or -curr.height)/factor
1162     hd = b2u and hd or -hd
1163     iy = b2u and iy or -iy
1164     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1165     for i=1,n do
1166         res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1167         dy = dy + hd + iy
1168     end
1169 end
1170 end
1171 end
1172 dy = dy + (b2u and vwidth or -vwidth)
1173 elseif curr.id == node.id"kern" then
1174     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1175 elseif curr.id == node.id"vlist" then
1176     dy = dy + (b2u and curr.depth or -curr.height)/factor
1177     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1178     dy = dy + (b2u and curr.height or -curr.depth)/factor
1179 elseif curr.id == node.id"hlist" then
1180     dy = dy + (b2u and curr.depth or -curr.height)/factor
1181     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1182     dy = dy + (b2u and curr.height or -curr.depth)/factor
1183 end
1184 curr = node.getnext(curr)
1185 end
1186 return res
1187 end
1188 function outline_horz (res, box, curr, xshift, yshift, discwd)
1189     local r2l = box.dir == "TRT"
1190     local dx = r2l and (discwd or box.width/factor) or 0
1191     local dirs = { { dir = r2l, dx = dx } }
1192     while curr do
1193         if curr.id == node.id"dir" then
1194             local sign, dir = curr.dir:match"()(...)"
1195             local level, newdir = curr.level, r2l
1196             if sign == "+" then
1197                 newdir = dir == "TRT"
1198                 if r2l ~= newdir then
1199                     local n = node.getnext(curr)
1200                     while n do
1201                         if n.id == node.id"dir" and n.level+1 == level then break end
1202                         n = node.getnext(n)
1203                     end
1204                     n = n or node.tail(curr)
1205                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1206                 end
1207                 dirs[level] = { dir = r2l, dx = dx }

```

```

1208     else
1209         local level = level + 1
1210         newdir = dirs[level].dir
1211         if r2l ~= newdir then
1212             dx = dirs[level].dx
1213         end
1214     end
1215     r2l = newdir
1216 elseif curr.char and curr.font and curr.font > 0 then
1217     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1218     local gid = ft.characters[curr.char].index or curr.char
1219     local scale = ft.size / factor / 1000
1220     local slant = (ft.slant or 0)/1000
1221     local extend = (ft.extend or 1000)/1000
1222     local squeeze = (ft.squeeze or 1000)/1000
1223     local expand = 1 + (curr.expansion_factor or 0)/1000000
1224     local xscale, yscale = scale * extend * expand, scale * squeeze
1225     dx = dx - (r2l and curr.width/factor*expand or 0)
1226     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1227     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1228     local vertical = ""
1229     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1230         if ft.shared.features.vertical then -- luatexko
1231             vertical = "rotated 90"
1232             local data = ft.characters[curr.char] or { }
1233             if ft.hb then
1234                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1235                 local charraise = (ft.luatexko_charraise or 0)/factor
1236                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1237             else
1238                 local cmds = data.commands or { {0,0}, {0,0} }
1239                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1240                 xpos, ypos = xpos + hoff, ypos + voff
1241             end
1242         elseif curr ~= box.head then -- luatexja
1243             vertical = "rotated 90"
1244             local en = ft.parameters.quad/factor/2
1245             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1246         end
1247     end
1248     local image
1249     if ft.format == "opentype" or ft.format == "truetype" then
1250         image = luamplib.glyph(curr.font, gid)
1251     else
1252         local name, scale = ft.name, 1
1253         local vf = font.read_vf(name, ft.size)
1254         if vf and vf.characters[gid] then
1255             local cmds = vf.characters[gid].commands or { }
1256             for _,v in ipairs(cmds) do

```

```

1257         if v[1] == "char" then
1258             gid = v[2]
1259         elseif v[1] == "font" and vf.fonts[v[2]] then
1260             name = vf.fonts[v[2]].name
1261             scale = vf.fonts[v[2]].size / ft.size
1262         end
1263     end
1264 end
1265 image = format("glyph %s of %q scaled %f", gid, name, scale)
1266 end
1267 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1268                     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1269 dx = dx + (r2l and 0 or curr.width/factor*expand)
1270 elseif curr.replace then
1271     local width = node.dimensions(curr.replace)/factor
1272     dx = dx - (r2l and width or 0)
1273     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1274     dx = dx + (r2l and 0 or width)
1275 elseif curr.id == node.id"rule" then
1276     local wd, ht, dp = getrulemetric(box, curr, true)
1277     if wd ~= 0 then
1278         local hd = ht + dp
1279         dx = dx - (r2l and wd or 0)
1280         if hd ~= 0 and curr.subtype == 0 then
1281             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1282         end
1283         dx = dx + (r2l and 0 or wd)
1284     end
1285 elseif curr.id == node.id"glue" then
1286     local width = node.effective_glue(curr, box)/factor
1287     dx = dx - (r2l and width or 0)
1288     if curr.leader then
1289         local curr, kind = curr.leader, curr.subtype
1290         if curr.id == node.id"rule" then
1291             local wd, ht, dp = getrulemetric(box, curr, true)
1292             local hd = ht + dp
1293             if hd ~= 0 then
1294                 wd = width
1295                 if wd ~= 0 and curr.subtype == 0 then
1296                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1297                 end
1298             end
1299         elseif curr.head then
1300             local wd = curr.width/factor
1301             if wd <= width then
1302                 local dx = r2l and dx+width or dx
1303                 local n, ix = 0, 0
1304                 if kind == 100 or kind == 103 then -- todo: gleaders
1305                     local adx = abs(dx-dirs[1].dx)

```

```

1306         local ndx = math.ceil(adx / wd) * wd
1307         local diff = ndx - adx
1308         n = math.floor((width-diff) / wd)
1309         dx = dx + (r2l and -diff-wd or diff)
1310     else
1311         n = math.floor(width / wd)
1312         if kind == 101 then
1313             local side = width % wd / 2
1314             dx = dx + (r2l and -side-wd or side)
1315         elseif kind == 102 then
1316             ix = width % wd / (n+1)
1317             dx = dx + (r2l and -ix-wd or ix)
1318         end
1319     end
1320     wd = r2l and -wd or wd
1321     ix = r2l and -ix or ix
1322     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1323     for i=1,n do
1324         res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1325         dx = dx + wd + ix
1326     end
1327 end
1328 end
1329 end
1330 dx = dx + (r2l and 0 or width)
1331 elseif curr.id == node.id"kern" then
1332     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1333 elseif curr.id == node.id"math" then
1334     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1335 elseif curr.id == node.id"vlist" then
1336     dx = dx - (r2l and curr.width/factor or 0)
1337     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1338     dx = dx + (r2l and 0 or curr.width/factor)
1339 elseif curr.id == node.id"hlist" then
1340     dx = dx - (r2l and curr.width/factor or 0)
1341     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1342     dx = dx + (r2l and 0 or curr.width/factor)
1343 end
1344 curr = node.getnext(curr)
1345 end
1346 return res
1347 end
1348 function luamplib.outlinetext (text)
1349     local fmt = process_tex_text(text)
1350     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1351     local box = texgetbox(id)
1352     local res = outline_horz({ }, box, box.head, 0, 0)
1353     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1354     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)

```

```

1355 end
1356 end
1357
    lua functions for mplib(uc)substring ... of ...
1358 function luamplib.getunicodegraphemes (s)
1359   local t = { }
1360   local graphemes = require'lua-uni-graphemes'
1361   for _, _, c in graphemes.graphemes(s) do
1362     table.insert(t, c)
1363   end
1364   return t
1365 end
1366 function luamplib.unicodesubstring (s,b,e,grph)
1367   local tt, t, step = { }
1368   if grph then
1369     t = luamplib.getunicodegraphemes(s)
1370   else
1371     t = { }
1372     for _, c in utf8.codes(s) do
1373       table.insert(t, utf8.char(c))
1374     end
1375   end
1376   if b <= e then
1377     b, step = b+1, 1
1378   else
1379     e, step = e+1, -1
1380   end
1381   for i = b, e, step do
1382     table.insert(tt, t[i])
1383   end
1384   s = table.concat(tt):gsub("'", "'&ditto'")
1385   return string.format("%s", s)
1386 end
1387
    METAPOST preambles
1388 luamplib.preambles = {
1389   preamble = [[
1390 boolean mplib ; mplib := true ;
1391 let dump = endinput ;
1392 let normalfontsize = fontsize;
1393 input %s ;
1394 ]],
1395   mplibcode = [[
1396 texscriptmode := 2;
1397 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1398 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1399 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1400 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;

```

```

1401 if known context_mlib:
1402   defaultfont := "cmtt10";
1403   let infont = normalinfont;
1404   let fontsize = normalfontsize;
1405   vardef thelabel@#(expr p,z) =
1406     if string p :
1407       thelabel@#(p infont defaultfont scaled defaultscale,z)
1408     else :
1409       p shifted (z + labeloffset*mfun_laboff@# -
1410         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1411         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1412     fi
1413   enddef;
1414 else:
1415   vardef texttext@# primary t = rawtexttext (t) enddef;
1416   def message expr t =
1417     if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1418   enddef;
1419   def withtransparency (expr a, t) =
1420     withprescript "tr_alternative=" & if numeric a: decimal fi a
1421     withprescript "tr_transparency=" & decimal t
1422   enddef;
1423   vardef ddecimal primary p =
1424     decimal xpart p & " " & decimal ypart p
1425   enddef;
1426   vardef boundingbox primary p =
1427     if (path p) or (picture p) :
1428       llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1429     else :
1430       origin
1431     fi -- cycle
1432   enddef;
1433 fi
1434 def resolvedcolor(expr s) =
1435   runscript("return luamplib.shadecolor('"&s&"')")
1436 enddef;
1437 def colordecimals primary c =
1438   if cmykcolor c:
1439     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1440     decimal yellowpart c & ":" & decimal blackpart c
1441   elseif rgbcolor c:
1442     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1443   elseif string c:
1444     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1445   else:
1446     decimal c
1447   fi
1448 enddef;
1449 def externalfigure primary filename =

```

```

1450 draw rawtexttext("\includegraphics{"& filename &}")
1451 enddef;
1452 def TEX = texttext enddef;
1453 def mplibtexcolor primary c =
1454   runscript("return luamplib.gettexcolor('"& c & "')")
1455 enddef;
1456 def mplibrgbtexcolor primary c =
1457   runscript("return luamplib.gettexcolor('"& c & "','rgb')")
1458 enddef;
1459 def mplibgraphictext primary t =
1460   begingroup;
1461   mplibgraphictext_ (t)
1462 enddef;
1463 def mplibgraphictext_ (expr t) text rest =
1464   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1465   fb, fc, dc, graphictextpic, alsoordoublepath;
1466   picture graphictextpic; graphictextpic := nullpicture;
1467   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1468   let scale = scaled;
1469   def fakebold primary c = hide(fb:=c;) enddef;
1470   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1471   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1472   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1473   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1474   addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1475   def fakebold primary c = enddef;
1476   let fillcolor = fakebold; let drawcolor = fakebold;
1477   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1478   image(draw runscript("return luamplib.graphictext([==["&t&"]==],",
1479     & decimal fb &,"'"& fc &","'"& dc &")") rest;)
1480 endgroup;
1481 enddef;
1482 def mplibglyph expr c of f =
1483   runscript (
1484     "return luamplib.glyph('"
1485     & if numeric f: decimal fi f
1486     & "',''"
1487     & if numeric c: decimal fi c
1488     & "')"
1489   )
1490 enddef;
1491 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1492 def mplibdrawglyph expr g =
1493   luamplib_tmp_num_ := 0;
1494   for item within g:
1495     fill pathpart item
1496     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1497   endfor
1498 enddef;

```

```

1499 let mplibfillglyph = mplibdrawglyph;
1500 def mplibstrokeglyph expr g =
1501   luamplib_tmp_num_ := 0;
1502   for item within g:
1503     draw pathpart item
1504     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1505   endfor
1506 enddef;
1507 def mplibfillandstrokeglyph expr g =
1508   luamplib_tmp_num_ := 0;
1509   for item within g:
1510     draw pathpart item withpostscript
1511     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1512   endfor
1513 enddef;
1514 def withmplibcolors (expr f, s) =
1515   runscript("return luamplib.fillandstrokecolor(' &
1516     if not string f: colordecimals fi f & '", ' &
1517     if not string s: colordecimals fi s & "')")
1518 enddef;
1519 def withmplibopacities (expr a, f, s) =
1520   withprescript "tr_alternative=" & if numeric a: decimal fi a
1521   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1522 enddef;
1523 def mplib_do_outline_text_set_b (text f) (text d) text r =
1524   def mplib_do_outline_options_f = f enddef;
1525   def mplib_do_outline_options_d = d enddef;
1526   def mplib_do_outline_options_r = r enddef;
1527 enddef;
1528 def mplib_do_outline_text_set_f (text f) text r =
1529   def mplib_do_outline_options_f = f enddef;
1530   def mplib_do_outline_options_r = r enddef;
1531 enddef;
1532 def mplib_do_outline_text_set_u (text f) text r =
1533   def mplib_do_outline_options_f = f enddef;
1534 enddef;
1535 def mplib_do_outline_text_set_d (text d) text r =
1536   def mplib_do_outline_options_d = d enddef;
1537   def mplib_do_outline_options_r = r enddef;
1538 enddef;
1539 def mplib_do_outline_text_set_r (text d) (text f) text r =
1540   def mplib_do_outline_options_d = d enddef;
1541   def mplib_do_outline_options_f = f enddef;
1542   def mplib_do_outline_options_r = r enddef;
1543 enddef;
1544 def mplib_do_outline_text_set_n text r =
1545   def mplib_do_outline_options_r = r enddef;
1546 enddef;
1547 def mplib_do_outline_text_set_p = enddef;

```

```

1548 def mplib_fill_outline_text =
1549   for n=1 upto mpliboutlinenum:
1550     i:=0;
1551     for item within mpliboutlinepic[n]:
1552       i:=i+1;
1553       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1554       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1555     endfor
1556   endfor
1557 enddef;
1558 def mplib_draw_outline_text =
1559   for n=1 upto mpliboutlinenum:
1560     for item within mpliboutlinepic[n]:
1561       draw pathpart item mplib_do_outline_options_d;
1562     endfor
1563   endfor
1564 enddef;
1565 def mplib_filldraw_outline_text =
1566   for n=1 upto mpliboutlinenum:
1567     i:=0;
1568     for item within mpliboutlinepic[n]:
1569       i:=i+1;
1570       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1571         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1572       else:
1573         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1574       fi
1575     endfor
1576   endfor
1577 enddef;
1578 vardef mpliboutlinetext@# (expr t) text rest =
1579   save kind; string kind; kind := str @#;
1580   save i; numeric i;
1581   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1582   def mplib_do_outline_options_d = enddef;
1583   def mplib_do_outline_options_f = enddef;
1584   def mplib_do_outline_options_r = enddef;
1585   runscript("return luamplib.outlinetext[==["&t&"]==]");
1586   image ( addto currentpicture also image (
1587     if kind = "f":
1588       mplib_do_outline_text_set_f rest;
1589       mplib_fill_outline_text;
1590     elseif kind = "d":
1591       mplib_do_outline_text_set_d rest;
1592       mplib_draw_outline_text;
1593     elseif kind = "b":
1594       mplib_do_outline_text_set_b rest;
1595       mplib_fill_outline_text;
1596       mplib_draw_outline_text;

```

```

1597 elseif kind = "u":
1598     mplib_do_outline_text_set_u rest;
1599     mplib_filldraw_outline_text;
1600 elseif kind = "r":
1601     mplib_do_outline_text_set_r rest;
1602     mplib_draw_outline_text;
1603     mplib_fill_outline_text;
1604 elseif kind = "p":
1605     mplib_do_outline_text_set_p;
1606     mplib_draw_outline_text;
1607 else:
1608     mplib_do_outline_text_set_n rest;
1609     mplib_fill_outline_text;
1610 fi;
1611 ) mplib_do_outline_options_r; )
1612 enddef ;
1613 def withmppattern primary p =
1614   withprescript "mplibpattern=" & if numeric p: decimal fi p
1615 enddef;
1616 def withpatternstroke expr a =
1617   withprescript "mplibpatternstroke=" & a
1618 enddef;
1619 primarydef t withpattern p =
1620   image(
1621     if cycle t:
1622       fill
1623     else:
1624       draw
1625     fi
1626     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1627 enddef;
1628 vardef mplibtransformmatrix (text e) =
1629   save t; transform t;
1630   t = identity e;
1631   runscript("luamplib.transformmatrix = {"
1632     & decimal xpart t & ","
1633     & decimal ypart t & ","
1634     & decimal xpart t & ","
1635     & decimal ypart t & ","
1636     & decimal xpart t & ","
1637     & decimal ypart t & ","
1638     & "}");
1639 enddef;
1640 primarydef p withmaskinggroup s =
1641   if picture p:
1642     image(
1643       draw p;
1644       draw center p withprescript "mplibfadestate=stop";
1645     )

```

```

1646 else:
1647   p withprescript "mplibfadestate=stop"
1648 fi
1649 withprescript "mplibfadetype=masking"
1650 withprescript "mplibmaskname=" & s
1651 enddef;
1652 def withmaskingbgcolor expr c =
1653   withprescript "mplibmaskingbgcolor=" & colordecimals c
1654 enddef;
1655 primarydef p withfademethod s =
1656   if picture p:
1657     image(
1658       draw p;
1659       draw center p withprescript "mplibfadestate=stop";
1660     )
1661   else:
1662     p withprescript "mplibfadestate=stop"
1663   fi
1664   withprescript "mplibfadetype=" & s
1665   hide(mplib_shade_step := 1;)
1666   withprescript "sh_color_a=1"
1667   withprescript "sh_color_b=0"
1668   withprescript "mplibfadebbox=" &
1669     decimal (xpart llcorner p -1/4) & ":" &
1670     decimal (ypart llcorner p -1/4) & ":" &
1671     decimal (xpart urcorner p +1/4) & ":" &
1672     decimal (ypart urcorner p +1/4)
1673 enddef;
1674 def withfadevector (expr a,b) =
1675   withprescript "mplibfadevector=" &
1676     decimal xpart a & ":" &
1677     decimal ypart a & ":" &
1678     decimal xpart b & ":" &
1679     decimal ypart b
1680 enddef;
1681 let withfadecenter = withfadevector;
1682 def withfaderadius (expr a,b) =
1683   withprescript "mplibfaderadius=" &
1684     decimal a & ":" &
1685     decimal b
1686 enddef;
1687 def withfadebbox (expr a,b) =
1688   withprescript "mplibfadebbox=" &
1689     decimal xpart a & ":" &
1690     decimal ypart a & ":" &
1691     decimal xpart b & ":" &
1692     decimal ypart b
1693 enddef;
1694 def withadjustbbox expr n =

```

```

1695 withprescript "mplibadjustbbox=" & decimal n
1696 enddef;
1697 primarydef p asgroup s =
1698   image(
1699     draw center p
1700     withprescript "mplibgroupbbox=" &
1701       decimal (xpart llcorner p -1/4) & ":" &
1702       decimal (ypart llcorner p -1/4) & ":" &
1703       decimal (xpart urcorner p +1/4) & ":" &
1704       decimal (ypart urcorner p +1/4)
1705     withprescript "gr_state=start"
1706     withprescript "gr_type=" & s;
1707     draw p withprescript "sh_in_xobj=yes";
1708     draw center p withprescript "gr_state=stop";
1709   )
1710 enddef;
1711 def withgroupbbox (expr a,b) =
1712   withprescript "mplibgroupbbox=" &
1713     decimal xpart a & ":" &
1714     decimal ypart a & ":" &
1715     decimal xpart b & ":" &
1716     decimal ypart b
1717 enddef;
1718 def withgroupname expr s =
1719   withprescript "mplibgroupname=" & s
1720 enddef;
1721 def usemplibgroup primary s =
1722   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s & "\endcsname}")
1723   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1724 enddef;
1725 path   mplib_shade_path ;
1726 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1727 numeric mplib_shade_fx, mplib_shade_fy ;
1728 numeric mplib_shade_lx, mplib_shade_ly ;
1729 numeric mplib_shade_nx, mplib_shade_ny ;
1730 numeric mplib_shade_dx, mplib_shade_dy ;
1731 numeric mplib_shade_tx, mplib_shade_ty ;
1732 primarydef p withshadingmethod m =
1733   p
1734   if picture p :
1735     withprescript "sh_operand_type=picture"
1736     if textual p or (length p > 1):
1737       withprescript "sh_transform=no"
1738       mplib_with_shade_method (boundingbox p, m)
1739     else:
1740       withprescript "sh_transform=yes"
1741       mplib_with_shade_method (pathpart p, m)
1742     fi
1743   else :

```

```

1744     withprescript "sh_transform=yes"
1745     mplib_with_shade_method (p, m)
1746   fi
1747 enddef;
1748 def mplib_with_shade_method (expr p, m) =
1749   hide(mplib_with_shade_method_analyze(p))
1750   withprescript "sh_domain=0 1"
1751   withprescript "sh_color=into"
1752   withprescript "sh_color_a=" & colordecimals white
1753   withprescript "sh_color_b=" & colordecimals black
1754   withprescript "sh_first=" & ddecimal point 0 of p
1755   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1756   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1757   if m = "linear" :
1758     withprescript "sh_type=linear"
1759     withprescript "sh_factor=1"
1760     withprescript "sh_center_a=" & ddecimal llcorner p
1761     withprescript "sh_center_b=" & ddecimal urcorner p
1762   elseif m = "coons":
1763     withprescript "sh_type=coons"
1764     withprescript "sh_transform=no"
1765   elseif m = "triangle":
1766     withprescript "sh_type=triangle"
1767     withprescript "sh_transform=no"
1768   elseif m = "lattice":
1769     withprescript "sh_type=lattice"
1770     withprescript "sh_transform=no"
1771   elseif m = "tensor":
1772     withprescript "sh_type=tensor"
1773     withprescript "sh_transform=no"
1774     withprescript "sh_tensor_path=" &
1775       ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1776       ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1777       ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1778       ddecimal 1/4[point 3 of p, point 1 of p]
1779   else :
1780     withprescript "sh_type=circular"
1781     withprescript "sh_factor=1.2"
1782     withprescript "sh_center_a=" & ddecimal center p
1783     withprescript "sh_center_b=" & ddecimal center p
1784     withprescript "sh_radius_a=" & decimal 0
1785     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1786   fi
1787 enddef;
1788 def withlatticeverticesperrow expr n =
1789   withprescript "sh_lattice_perrow=" & decimal n
1790 enddef;
1791 def withlatticeverticesdata (text t) =
1792   hide(luamplib_tmp_num_ := 0;)

```

```

1793 withprescript "sh_lattice_data=" &
1794 for item = t:
1795   if odd(incr luamplib_tmp_num_):
1796     if pair item: ddecimal fi item & " " &
1797     fi
1798   endfor ""
1799   hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1800   for item = t:
1801     if not odd(incr luamplib_tmp_num_):
1802       withshadingstep( withshadingcolors(item,item) )
1803     fi
1804   endfor
1805 enddef;
1806 def withtrianglepatchinit (expr p, a, b, c) =
1807   if string p:
1808     withtrianglepatchnext(0, p , a)
1809     withtrianglepatchnext(0, "", b)
1810     withtrianglepatchnext(0, "", c)
1811   else:
1812     withtrianglepatchnext(0, point 0 of p, a)
1813     withtrianglepatchnext(0, point 1 of p, b)
1814     withtrianglepatchnext(0, point 2 of p, c)
1815   fi
1816 enddef;
1817 def withtrianglepatchnext (expr f, p, a) =
1818   withshadingstep(
1819     withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1820     withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1821     if string p: p else: ddecimal p fi
1822     withshadingcolors (a, a)
1823   )
1824 enddef;
1825 def withcoonspatchinit (expr p, a, b, c, d) =
1826   withshadingstep(
1827     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1828     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1829     if string p: p
1830     else:
1831       ddecimal point      0 of p & " " &
1832       ddecimal postcontrol 0 of p & " " &
1833       ddecimal precontrol  1 of p & " " &
1834       ddecimal point      1 of p & " " &
1835       ddecimal postcontrol 1 of p & " " &
1836       ddecimal precontrol  2 of p & " " &
1837       ddecimal point      2 of p & " " &
1838       ddecimal postcontrol 2 of p & " " &
1839       ddecimal precontrol  3 of p & " " &
1840       ddecimal point      3 of p & " " &
1841       ddecimal postcontrol 3 of p & " " &

```

```

1842         ddecimal precontrol 0 of p
1843     fi
1844     withshadingcolors (a, b)
1845 )
1846 withshadingstep ( withshadingcolors (c, d) )
1847 enddef;
1848 def withtensorpatchinit (expr p, q, a, b, c, d) =
1849     withshadingstep(
1850         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1851         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1852             if string p: p & " " & q
1853             else:
1854                 ddecimal point 0 of p & " " &
1855                 ddecimal postcontrol 0 of p & " " &
1856                 ddecimal precontrol 1 of p & " " &
1857                 ddecimal point 1 of p & " " &
1858                 ddecimal postcontrol 1 of p & " " &
1859                 ddecimal precontrol 2 of p & " " &
1860                 ddecimal point 2 of p & " " &
1861                 ddecimal postcontrol 2 of p & " " &
1862                 ddecimal precontrol 3 of p & " " &
1863                 ddecimal point 3 of p & " " &
1864                 ddecimal postcontrol 3 of p & " " &
1865                 ddecimal precontrol 0 of p & " " &
1866                 ddecimal point 0 of q & " " &
1867                 ddecimal point 1 of q & " " &
1868                 ddecimal point 2 of q & " " &
1869                 ddecimal point 3 of q
1870             fi
1871         withshadingcolors (a, b)
1872     )
1873 withshadingstep ( withshadingcolors (c, d) )
1874 enddef;
1875 def withcoonspatchnext (expr f, p, a, b) =
1876     withshadingstep(
1877         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1878         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1879             if string p: p
1880             else:
1881                 ddecimal postcontrol 1 of p & " " &
1882                 ddecimal precontrol 2 of p & " " &
1883                 ddecimal point 2 of p & " " &
1884                 ddecimal postcontrol 2 of p & " " &
1885                 ddecimal precontrol 3 of p & " " &
1886                 ddecimal point 3 of p & " " &
1887                 ddecimal postcontrol 3 of p & " " &
1888                 ddecimal precontrol 0 of p
1889             fi
1890         withshadingcolors (a, b)

```

```

1891 )
1892 enddef;
1893 def withtensorpatchnext (expr f, p, q, a, b) =
1894   withshadingstep(
1895     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1896     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1897       if string p: p & " " & q
1898       else:
1899         ddecimal postcontrol 1 of p & " " &
1900         ddecimal precontrol 2 of p & " " &
1901         ddecimal point 2 of p & " " &
1902         ddecimal postcontrol 2 of p & " " &
1903         ddecimal precontrol 3 of p & " " &
1904         ddecimal point 3 of p & " " &
1905         ddecimal postcontrol 3 of p & " " &
1906         ddecimal precontrol 0 of p & " " &
1907         ddecimal point 0 of q & " " &
1908         ddecimal point 1 of q & " " &
1909         ddecimal point 2 of q & " " &
1910         ddecimal point 3 of q
1911       fi
1912     withshadingcolors (a, b)
1913   )
1914 enddef;
1915 def mplib_with_shade_method_analyze(expr p) =
1916   mplib_shade_path := p ;
1917   mplib_shade_step := 1 ;
1918   mplib_shade_fx := xpart point 0 of p ;
1919   mplib_shade_fy := ypart point 0 of p ;
1920   mplib_shade_lx := mplib_shade_fx ;
1921   mplib_shade_ly := mplib_shade_fy ;
1922   mplib_shade_nx := 0 ;
1923   mplib_shade_ny := 0 ;
1924   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1925   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1926   for i=1 upto length(p) :
1927     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1928     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1929     if mplib_shade_tx > mplib_shade_dx :
1930       mplib_shade_nx := i + 1 ;
1931       mplib_shade_lx := xpart point i of p ;
1932       mplib_shade_dx := mplib_shade_tx ;
1933     fi ;
1934     if mplib_shade_ty > mplib_shade_dy :
1935       mplib_shade_ny := i + 1 ;
1936       mplib_shade_ly := ypart point i of p ;
1937       mplib_shade_dy := mplib_shade_ty ;
1938     fi ;
1939   endfor ;

```

```

1940 enddef;
1941 vardef mplib_max_radius(expr p) =
1942   max (
1943     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1944     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1945     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1946     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1947   )
1948 enddef;
1949 def withshadingstep (text t) =
1950   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1951   withprescript "sh_step=" & decimal mplib_shade_step
1952   t
1953 enddef;
1954 let withfadestep = withshadingstep;
1955 def withshadingradius expr a =
1956   withprescript "sh_radius_a=" & decimal (xpart a)
1957   withprescript "sh_radius_b=" & decimal (ypart a)
1958 enddef;
1959 def withshadingorigin expr a =
1960   withprescript "sh_center_a=" & ddecimal a
1961   withprescript "sh_center_b=" & ddecimal a
1962 enddef;
1963 def withshadingvector expr a =
1964   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1965   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1966 enddef;
1967 def withshadingdirection expr a =
1968   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1969   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1970 enddef;
1971 def withshadingtransform expr a =
1972   withprescript "sh_transform=" & a
1973 enddef;
1974 def withshadingcenter expr a =
1975   withprescript "sh_center_a=" & ddecimal (
1976     center mplib_shade_path shifted (
1977       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1978       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1979     )
1980   )
1981 enddef;
1982 def withshadingcenters (expr a, b) =
1983   withprescript "sh_center_a=" & ddecimal a
1984   withprescript "sh_center_b=" & ddecimal b
1985   withshadingtransform "no"
1986   withshadingfactor 1
1987 enddef;
1988 let withshadingpoints = withshadingcenters;

```

```

1989 def withshadingextend (expr a, b) =
1990   withprescript "sh_extend=" &
1991     if a: "true" else: "false" fi & " " &
1992     if b: "true" else: "false" fi
1993 enddef;
1994 let withfadeextend = withshadingextend;
1995 def withshadingdomain expr d =
1996   withprescript "sh_domain=" & ddecimal d
1997 enddef;
1998 def withshadingfactor expr f =
1999   withprescript "sh_factor=" & decimal f
2000 enddef;
2001 def withshadingfraction expr a =
2002   if mplib_shade_step > 0 :
2003     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
2004   fi
2005 enddef;
2006 let withfadefraction = withshadingfraction;
2007 def withshadingcolors (expr a, b) =
2008   if mplib_shade_step > 0 :
2009     withprescript "sh_color=into"
2010     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
2011     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
2012   else :
2013     withprescript "sh_color=into"
2014     withprescript "sh_color_a=" & colordecimals a
2015     withprescript "sh_color_b=" & colordecimals b
2016   fi
2017 enddef;
2018 let withfadeopacity = withshadingcolors;
2019 def withshadingstroke expr a =
2020   withprescript "sh_stroking=" & a
2021 enddef;
2022 def withshadingmatrix expr s =
2023   withprescript "sh_matrix=" & s
2024 enddef;
2025 let withfadematrix = withshadingmatrix;
2026 def mpliblength primary t =
2027   runscript("return utf8.len[==[" & t & "]==]")
2028 enddef;
2029 def mplibsubstring expr p of t =
2030   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
2031     & decimal xpart p & ",",
2032     & decimal ypart p & ")")
2033 enddef;
2034 def mplibbuclength primary t =
2035   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
2036 enddef;
2037 def mplibbuclsubstring expr p of t =

```

```

2038  runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
2039    & decimal xpart p & ",",
2040    & decimal ypart p & ",true)")
2041 enddef;
2042 ]],
2043 legacyverbatimtex = [[
2044 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
2045 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
2046 let VerbatimTeX = specialVerbatimTeX;
2047 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
2048   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
2049 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
2050   "runscript(" &ditto&
2051   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
2052   "luamplib.in_the_fig=false" &ditto& ");";
2053 ]],
2054 texttextlabel = [[
2055 let luampliboriginalinfont = infont;
2056 primarydef s infont f =
2057   if (s < char 32)
2058     or (s = char 35) % #
2059     or (s = char 36) % $
2060     or (s = char 37) % %
2061     or (s = char 38) % &
2062     or (s = char 92) % \
2063     or (s = char 94) % ^
2064     or (s = char 95) % _
2065     or (s = char 123) % {
2066     or (s = char 125) % }
2067     or (s = char 126) % ~
2068     or (s = char 127) :
2069     s luampliboriginalinfont f
2070   else :
2071     rawtexttext(s)
2072   fi
2073 enddef;
2074 def fontsize expr f =
2075   begingroup
2076   save size; numeric size;
2077   size := mplibdimen("1em");
2078   if size = 0: 10pt else: size fi
2079   endgroup
2080 enddef;
2081 ]],
2082 }
2083

```

process_mplibcode

For externalize feature, the code for which is located at the end of lua file.

```

2084 luamplib.externalize = luamplib.externalize or { }
2085 local externalize = luamplib.externalize
2086

```

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

```

2087 luamplib.verbatiminput = false
2088 luamplib.everymplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2089 luamplib.everyendmplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2090 function luamplib.process_mplibcode (data, instancename)
2091   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2092   if luamplib.legacyverbatim then
2093     luamplib.figid, tex_code_pre_mplib = 1, {}
2094   end
2095   local everymplib = luamplib.everymplib[instancename]
2096   local everyendmplib = luamplib.everyendmplib[instancename]
2097   data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
2098   :gsub("\r", "\n")

```

These lines are needed for `mplibverbatim` mode.

```

2099   if luamplib.verbatiminput then
2100     data = data:gsub("\mpcolor%+{.-}%b{+}", "mplibcolor(\"%1\")")
2101     :gsub("\mpdim%+{.-}%b{+}", "mplibdimen(\"%1\")")
2102     :gsub("\mpdim%+{.-}%a{+}", "mplibdimen(\"%1\")")
2103     data = scan_mpcode(data, replace_texblock)
2104   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.

```

2105   local t = { } -- to store btex, verbatimtex, string
2106   local function store (str)
2107     t[#t+1] = str
2108     return #t
2109   end
2110   data = scan_mpcode(data, {
2111     btex = function(str)
2112       return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", store(str)) -- space
2113     end,
2114     verbatimtex = function(str)
2115       return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", store(str)) -- semicolon
2116     end,
2117     str = function(str)
2118       return format("\\" .. unexpanded{!l!u!a!%s!m!p!l!}''', store(str))
2119     end,
2120     comment = function() return "" end,
2121     escapepercent = true,
2122   })
2123   run_tex_code(format("\mplibtmptoks\\expandafter{\\expanded{%s}}", data))
2124   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

2125 :gsub("##", "#")
2126 :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2127 end
2128
2129 if externalize.running then
2130   if externalize.figure(data) then return end
2131 end
2132
2133 process(data, instancename)
2134 end
2135

```

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.

```

2136 local figcontents = { post = { } }
2137 local function put2output(a,...)
2138   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2139 end
2140 local function pdf_startfigure(n,llx,lly,urx,ury)
2141   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2142 end
2143 local function pdf_stopfigure()
2144   put2output("\mplibstoptoPDF")
2145 end

```

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

```

2146 local function pdf_literalcode (...)
2147   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2148 end
2149 local start_pdf_code = pdfmode
2150 and function() pdf_literalcode"q" end
2151 or function() put2output"\special{pdf:bcontent}" end
2152 local stop_pdf_code = pdfmode
2153 and function() pdf_literalcode"Q" end
2154 or function() put2output"\special{pdf:econtent}" end
2155

```

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

```

2156 local function put_tex_boxes (object,prescript)
2157   local box = prescript.mplibtexboxid:explode":"
2158   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2159   if n and tw and th then
2160     local op = object.path
2161     local first, second, fourth = op[1], op[2], op[4]
2162     local tx, ty = first.x_coord, first.y_coord
2163     local sx, rx, ry, sy = 1, 0, 0, 1
2164     if tw ~= 0 then
2165       sx = (second.x_coord - tx)/tw
2166       rx = (second.y_coord - ty)/tw

```

```

2167     if sx == 0 then sx = 0.00001 end
2168 end
2169 if th ~= 0 then
2170     sy = (fourth.y_coord - ty)/th
2171     ry = (fourth.x_coord - tx)/th
2172     if sy == 0 then sy = 0.00001 end
2173 end
2174

```

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

```

2175     local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2176     put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2177 end
2178 end
2179

```

Colors

```

2180 local function do_preobj_CR(object,prescript)
2181     if object.postscript == "collect" then return end
2182     local override = prescript and prescript.mpliboverridecolor
2183     if override then
2184         pdf_literalcode(override)
2185         if not pdfmode and override:find" [cC][sS] " then
2186             local name1 = override:match("^/(.-) cs ")
2187             local name2 = override:match(" /(.-) CS ")
2188             if name1 then
2189                 texpriint(ccexplat,
2190                     "\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2191                     name1, "}{\pdf_object_ref:n{", name1, "}}"
2192                 )
2193             end
2194             if name2 and name1 ~= name2 then
2195                 texpriint(ccexplat,
2196                     "\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2197                     name2, "}{\pdf_object_ref:n{", name2, "}}"
2198                 )
2199             end
2200         end
2201     else
2202         local cs = object.color
2203         if cs and #cs > 0 then
2204             pdf_literalcode(luamplib.colorconverter(cs))
2205         end
2206     end
2207 end
2208

```

For transparency, shading, fading, and pattern

```

2209 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2210 local pdfobjs, pdfetcs = {}, {}

```

```

2211 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2212 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2213 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
2214 local update_pdfobjs
2215 if pdfmode then
2216   function update_pdfobjs (os, stream)
2217     local key = os
2218     if stream then key = key..stream end
2219     local on = key and pdfobjs[key]
2220     if on then
2221       return on,false
2222     end
2223     if stream then
2224       on = pdf.immediateobj("stream",stream,os)
2225     elseif os then
2226       on = pdf.immediateobj(os)
2227     else
2228       on = pdf.reserveobj()
2229     end
2230     if key then
2231       pdfobjs[key] = on
2232     end
2233     return on,true
2234   end
2235 else
2236   function update_pdfobjs (os, stream, hex)
2237     local key = os
2238     if stream then key = key..stream end
2239     local on = key and pdfobjs[key]
2240     if on then
2241       return on,false
2242     end
2243     on = pdfetcs.cnt or 1
2244     if stream then
2245       if hex then
2246         texsprint(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2247       else
2248         texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2249       end
2250     elseif os then
2251       texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2252     else
2253       texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2254     end
2255     pdfetcs.cnt = on + 1
2256     if key then
2257       pdfobjs[key] = on
2258     end
2259     return on,true

```

```

2260 end
2261 end
2262 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2263 if pdfmode then
2264   pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2265   local getpagers = pdfetcs.getpagers
2266   local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2267   local initialize_resources = function (name)
2268     local tabname = format("%s_res",name)
2269     pdfetcs[tabname] = { }
2270     if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
2271       local obj = pdf.reserveobj()
2272       setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2273       luatexbase.add_to_callback("finish_pdffile", function()
2274         pdf.immediateobj(obj, format("<<s>>", tableconcat(pdfetcs[tabname])))
2275       end,
2276       format("luamplib.%s.finish_pdffile",name))
2277     end
2278   end
2279   pdfetcs.fallback_update_resources = function (name, res)
2280     local tabname = format("%s_res",name)
2281     if not pdfetcs[tabname] then
2282       initialize_resources(name)
2283     end
2284     if luatexbase.callbacktypes.finish_pdffile then
2285       local t = pdfetcs[tabname]
2286       t[#t+1] = res
2287     else
2288       local tpr, n = getpagers() or "", 0
2289       tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2290       if n == 0 then
2291         tpr = format("%s/%s<<s>>", tpr, name, res)
2292       end
2293       setpagers(tpr)
2294     end
2295   end
2296 else
2297   texsprint (
2298     "\\luamplibatfirstshipout{",
2299     "\\special{pdf:obj @MPlibTr<<>>}",
2300     "\\special{pdf:obj @MPlibSh<<>>}",
2301     "\\special{pdf:obj @MPlibCS<<>>}",
2302     "\\special{pdf:obj @MPlibPt<<>>}}")
2303 )
2304 pdfetcs.fallback_update_resources = function (name,res,obj)
2305   texsprint("\\special{pdf:put ", obj, " <<", res, ">>}")
2306   local tabname = format("%s_res",name)
2307   if not pdfetcs[tabname] then
2308     texsprint("\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}")

```

```

2309     pdfetcs[tabname] = { }
2310     end
2311     tableinsert(pdfetcs[tabname], res)
2312 end
2313 end
2314

```

Transparency

```

2315 local function add_extgs_resources (on, new)
2316   local key = format("MPLibTr%s", on)
2317   if new then
2318     local val = format(pdfetcs.resfmt, on)
2319     if pdfmanagement then
2320       texsprintf (
2321         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2322       )
2323     else
2324       local tr = format("/%s %s", key, val)
2325       if is_defined(pdfetcs.pgfbxextgs) then
2326         texsprintf ( "\\csname ", pdfetcs.pgfbxextgs, "\\endcsname{", tr, "}" )
2327       elseif is_defined"TRP@list" then
2328         texsprintf(catat11,
2329           [[\if@files\immediate\write\@auxout{]],
2330           [[\string\g@addto@macro\string\TRP@list{]],
2331           tr,
2332           [[}]\fi]]
2333         )
2334         if not get_macro"TRP@list":find(tr) then
2335           texsprintf(catat11,[[\global\TRP@reruntrue]])
2336         end
2337       else
2338         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2339       end
2340     end
2341   end
2342   return key
2343 end
2344
2345 local do_preobj_TR
2346 do
2347   local transparency_modes = {
2348     [0] = "Normal",
2349     "Normal",      "Multiply",    "Screen",      "Overlay",
2350     "SoftLight",   "HardLight",   "ColorDodge",  "ColorBurn",
2351     "Darken",      "Lighten",    "Difference",  "Exclusion",
2352     "Hue",         "Saturation", "Color",       "Luminosity",
2353     "Compatible",
2354     normal        = "Normal",    multiply       = "Multiply",   screen        = "Screen",
2355     overlay       = "Overlay",   softlight     = "SoftLight",  hardlight     = "HardLight",

```

```

2356   colordodge = "ColorDodge", colorburn = "ColorBurn", darken = "Darken",
2357   lighten = "Lighten", difference = "Difference", exclusion = "Exclusion",
2358   hue = "Hue", saturation = "Saturation", color = "Color",
2359   luminosity = "Luminosity", compatible = "Compatible",
2360 }
2361 local full_opacity
2362 local function get_full_opacity()
2363   local on, new = update_pdfobjs"<</BM/Normal/ca 1/CA 1/AIS false>>"
2364   local key = add_extgs_resources(on,new)
2365   full_opacity = format("/%s gs",key)
2366   return full_opacity
2367 end
2368 function do_preobj_TR(object,prescript)
2369   if object.postscript == "collect" then return end
2370   local opa = prescript and prescript.tr_transparency
2371   if not opa then return end
2372
2373   local key, on, os, new
2374   local mode = prescript.tr_alternative or 1
2375   mode = transparency_modes[tonumber(mode) or mode:lower()]
2376   if not mode then
2377     mode = prescript.tr_alternative
2378     warn("unsupported blend mode: '%s'", mode)
2379   end
2380   opa = opa:explode":"
2381   for i,v in ipairs(opa) do
2382     opa[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2383   end
2384   os = format("<</BM/%s/ca %s/CA %s/AIS false>>",mode,opa[1],opa[2] or opa[1])
2385   on, new = update_pdfobjs(os)
2386   key = add_extgs_resources(on,new)
2387   pdf_literalcode("/%s gs",key)
2388   return full_opacity or get_full_opacity()
2389 end
2390 end
2391

```

Shading with *metafun* format.

```

2392 local function add_shading_resources (on, new)
2393   if new then
2394     local key, val = format("MPLibSh%s", on), format(pdfetcs.resfmt, on)
2395     if pdfmanagement then
2396       texsprintf (
2397         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2398       )
2399     else
2400       local res = format("/%s %s", key, val)
2401       pdfetcs.fallback_update_resources("Shading",res,"@MPLibSh")
2402     end

```

```

2403 end
2404 end
2405 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2406   for _, v in ipairs{ca, cb} do
2407     for i, vv in ipairs(v) do
2408       for ii, vvv in ipairs(vv) do
2409         v[i][ii] = tonumber(vvv) and format("%.3f", vvv) or vvv
2410       end
2411     end
2412   end
2413   local fun2fmt, os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2414   if steps > 1 then
2415     local list, bounds, encode = { }, { }, { }
2416     for i=1, steps do
2417       if i < steps then
2418         bounds[i] = format("%.3f", fractions[i] or 1)
2419       end
2420       encode[2*i-1] = 0
2421       encode[2*i] = 1
2422       os = fun2fmt:format(domain, tableconcat(ca[i], ' '), tableconcat(cb[i], ' '))
2423       :gsub(decimals, rmzeros)
2424       list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2425     end
2426     os = tableconcat {
2427       "<</FunctionType 3",
2428       format("/Bounds[%s]", tableconcat(bounds, ' ')),
2429       format("/Encode[%s]", tableconcat(encode, ' ')),
2430       format("/Functions[%s]", tableconcat(list, ' ')),
2431       format("/Domain[%s]>>", domain),
2432     } :gsub(decimals, rmzeros)
2433   else
2434     os = fun2fmt:format(domain, tableconcat(ca[1], ' '), tableconcat(cb[1], ' '))
2435     :gsub(decimals, rmzeros)
2436   end
2437   local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2438   os = tableconcat {
2439     format("<</ShadingType %i", shtype),
2440     format("/ColorSpace %s", colorspace),
2441     format("/Function %s", objref),
2442     format("/Coords[%s]", coordinates),
2443     format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2444   } :gsub(decimals, rmzeros)
2445   local on, new = update_pdfobjs(os)
2446   add_shading_resources(on, new)
2447   return on
2448 end
2449
2450 local function get_mp_matrix (matrix)
2451   process(("mplibtransformmatrix(%s);"):format(matrix), "@mplibtransformmatrix")

```

```

2452 return luamplib.transformmatrix
2453 end
2454
2455 local do_preobj_SH
2456 do
2457 pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2458     run_tex_code({
2459         [[\color_model_new:nnn]],
2460         format("{mplibcolorspace_%s}", names:gsub(",","_")),
2461         format("{DeviceN}{names={%s}}", names),
2462         [[\mplibtmptoks\expanded{{\pdf_object_ref_last:}}]],
2463     }, ccexplat)
2464     local colorspace = texgettoks"mplibtmptoks"
2465     t[names] = colorspace
2466     return colorspace
2467 end })
2468 local function color_normalize(ca,cb)
2469     if #cb == 1 then
2470         if #ca == 4 then
2471             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2472         else -- #ca = 3
2473             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2474         end
2475     elseif #cb == 3 then -- #ca == 4 : rgb to cmyk
2476         local c, m, y = 1-cb[1], 1-cb[2], 1-cb[3]
2477         local k = math.min(c, m, y)
2478         if k == 1 then
2479             c, m, y = 0, 0, 0
2480         else
2481             c, m, y = (c-k)/(1-k), (m-k)/(1-k), (y-k)/(1-k)
2482         end
2483         cb[1], cb[2], cb[3], cb[4] = c, m, y, k
2484     end
2485 end
2486 local function write_mesh_objs (shtype, colorspace, stream, devicen, perrow)
2487     local cc = colorspace == "/DeviceCMYK" and 4
2488         or colorspace == "/DeviceRGB" and 3
2489         or devicen or 1
2490     local dict = format(
2491         "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2492         shtype,
2493         perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2494         colorspace,
2495         (" 0 1"):rep(cc))
2496     local on, new
2497     if pdfmode then
2498         on, new = update_pdfobjs(dict, stream)
2499     else
2500         stream = format((" %02X"):rep(stream:len()), stream:byte(1,stream:len()))

```

```

2501     on, new = update_pdfobjs(dict, stream, true) -- hex is true
2502 end
2503 add_shading_resources(on, new)
2504 return on
2505 end
2506 local function colors_ff (colors)
2507     local t, devicen = { }, { }
2508     for i,v in ipairs(colors) do
2509         t[i] = { }
2510         for _,vv in ipairs(v) do
2511             local tt = tableconcat(vv," "):explode()
2512             for _,vzv in ipairs(tt) do
2513                 tableinsert(t[i], string.char(math.floor(vzv * 0xFF)))
2514             end
2515             devicen = #tt
2516         end
2517     end
2518     return t, devicen
2519 end
2520 local function coords_ffff (coords)
2521     local Xt, Yt = { }, { }
2522     for i,v in ipairs(coords) do
2523         for ii,vv in ipairs(v) do
2524             local t = ii % 2 == 1 and Xt or Yt
2525             t[#t+1] = tonumber(vv)
2526         end
2527     end
2528     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2529     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2530     local wd, ht = xmax - xmin, ymax - ymin
2531     for i,v in ipairs(coords) do
2532         for ii,vv in ipairs(v) do
2533             local min = ii % 2 == 1 and xmin or ymin
2534             local dim = ii % 2 == 1 and wd or ht
2535             coords[i][ii] = string.pack(">H", math.floor((vv - min)/dim * 0xFFFF ))
2536         end
2537     end
2538     return coords, xmin, ymin, wd, ht
2539 end
2540 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2541     local perrow = prescript.sh_lattice_perrow or 2
2542     local steps = tonumber(prescript.sh_step) or 0
2543     local coords, colors = { }, { }
2544     if steps < 4 then
2545         local path = object.path
2546         for _,i in ipairs{1,2,4,3} do
2547             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2548         end
2549         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}}, {{1,1,0}} }

```

```

2550     colorspace = "/DeviceRGB"
2551 else
2552     local t = prescript.sh_lattice_data:explode()
2553     for i = 1, #t, 2 do
2554         coords[#coords+1] = { t[i], t[i+1] }
2555     end
2556     for i = 1, steps do
2557         if not ca[i] then err"colorspace mismatch?" end
2558         colors[#colors+1] = { ca[i] }
2559     end
2560 end
2561 if #coords % perrow ~= 0 then err"vertices_per_row mismatches lattice_coords" end
2562
2563 local stream, xmin, ymin, wd, ht, devicen = { }
2564 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2565 colors, devicen = colors_ff(colors)
2566 for i = 1, #coords do
2567     stream[#stream+1] = tableconcat(coords[i])
2568     stream[#stream+1] = tableconcat(colors[i])
2569 end
2570
2571 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2572 local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2573 return on, matrix
2574 end
2575 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2576     local steps = tonumber(prescript.sh_step) or 0
2577     local coords, colors, edges = { }, { }, { }
2578     if steps < 3 then
2579         local path = object.path
2580         for i = 1, 3 do
2581             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2582         end
2583         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}} }
2584         edges = { 0, 0, 0 }
2585         colorspace = "/DeviceRGB"
2586     else
2587         for i = 1, steps do
2588             local t = prescript["sh_triangle_vertex_" .. i]:explode()
2589             if #t == 2 then
2590                 coords[#coords+1] = { t[1], t[2] }
2591             elseif #t == 6 then
2592                 coords[#coords+1] = { t[1], t[2] }
2593                 coords[#coords+1] = { t[3], t[4] }
2594                 coords[#coords+1] = { t[5], t[6] }
2595             end
2596             if not ca[i] then err"colorspace mismatch?" end
2597             colors[#colors+1] = { ca[i] }
2598             edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])

```

```

2599     end
2600 end
2601
2602 local stream, xmin, ymin, wd, ht, devicen = { }
2603 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2604 colors, devicen = colors_ff(colors)
2605 for i = 1, #coords do
2606     stream[#stream+1] = string.char(edges[i])
2607     stream[#stream+1] = tableconcat(coords[i])
2608     stream[#stream+1] = tableconcat(colors[i])
2609 end
2610
2611 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2612 local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2613 return on, matrix
2614 end
2615 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2616     local tensor = prescript.sh_type == "tensor"
2617     local steps = tonumber(prescript.sh_step) or 0
2618     local coords, colors, edges = { }, { }, { }
2619     if steps < 2 then
2620         local path, t = object.path, { }
2621         for i = 1, 4 do
2622             t[#t+1] = path[i].x_coord
2623             t[#t+1] = path[i].y_coord
2624             t[#t+1] = path[i].right_x
2625             t[#t+1] = path[i].right_y
2626             local j = i == 4 and 1 or i+1
2627             t[#t+1] = path[j].left_x
2628             t[#t+1] = path[j].left_y
2629         end
2630         if tensor then
2631             t = table.move(prescript.sh_tensor_path:explode(), 1, 8, #t+1, t)
2632         end
2633         coords = { t }
2634         colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2635         edges = { 0 }
2636         colorspace = "/DeviceRGB"
2637     else
2638         for i = 1, steps do
2639             local edge = tonumber(prescript["sh_coons_edge_"..i])
2640             if edge then
2641                 edges[#edges+1] = edge
2642                 coords[#coords+1] = prescript["sh_coons_path_"..i]:explode()
2643                 if edge == 0 then
2644                     if not (ca[i] and cb[i] and ca[i+1] and cb[i+1]) then err"colorspace mismatch?" end
2645                     colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2646                 else
2647                     if not (ca[i] and cb[i]) then err"colorspace mismatch?" end

```

```

2648         colors[#colors+1] = { ca[i], cb[i] }
2649     end
2650 end
2651 end
2652 end
2653
2654 local stream, xmin, ymin, wd, ht, devicen = { }
2655 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2656 colors, devicen = colors_ff(colors)
2657 for i = 1, #coords do
2658     stream[#stream+1] = string.char(edges[i])
2659     stream[#stream+1] = tableconcat(coords[i])
2660     stream[#stream+1] = tableconcat(colors[i])
2661 end
2662
2663 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2664 local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2665 return on, matrix
2666 end
2667 function do_preobj_SH(object, prescript)
2668     local shade_no
2669     local sh_type = prescript and prescript.sh_type
2670     if not sh_type then return end
2671
2672     local domain = prescript.sh_domain or "0 1"
2673     local centera = (prescript.sh_center_a or "0 0"):explode()
2674     local centerb = (prescript.sh_center_b or "0 0"):explode()
2675     local transform = prescript.sh_transform == "yes"
2676     local sx,sy,sr,dx,dy = 1,1,1,0,0
2677     if transform then
2678         local first = (prescript.sh_first or "0 0"):explode()
2679         local setx = (prescript.sh_set_x or "0 0"):explode()
2680         local sety = (prescript.sh_set_y or "0 0"):explode()
2681         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2682         if x ~= 0 and y ~= 0 then
2683             local path = object.path
2684             local path1x = path[1].x_coord
2685             local path1y = path[1].y_coord
2686             local path2x = path[x].x_coord
2687             local path2y = path[y].y_coord
2688             local dxa = path2x - path1x
2689             local dya = path2y - path1y
2690             local dxb = setx[2] - first[1]
2691             local dyb = sety[2] - first[2]
2692             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2693                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
2694                 sy = dya / dyb ; if sy < 0 then sy = - sy end
2695                 sr = math.sqrt(sx^2 + sy^2)
2696                 dx = path1x - sx*first[1]

```

```

2697         dy = path1y - sy*first[2]
2698     end
2699 end
2700 end
2701 local ca, cb, colorspace, steps, fractions
2702 ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode":" }
2703 cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode":" }
2704 steps = tonumber(prescript.sh_step) or 1
2705 if steps > 1 then
2706     fractions = { prescript.sh_fraction_1 or 0 }
2707     for i=2,steps do
2708         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2709         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode":"
2710         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode":"
2711     end
2712 end
2713 if prescript.mpllib_spotcolor then
2714     ca, cb = { }, { }
2715     local names, pos, objref = { }, -1, ""
2716     local script = object.prescript:explode"\13+"
2717     for i=#script,1,-1 do
2718         local name, value
2719         if script[i]:find"mpllib_spotcolor" then
2720             local t = script[i]:explode"="[2]:explode":"
2721             value, objref, name = t[1], t[2], t[3]
2722         elseif script[i]:find"mpllib_textcolor" then
2723             local t = script[i]:explode"="[2]:explode"!"
2724             name, value = t[1], (t[2] or 100)/100
2725         end
2726         if name and value then
2727             if not names[name] then
2728                 pos = pos+1
2729                 names[name] = pos
2730                 names[#names+1] = name
2731             end
2732             local t = { }
2733             for j=1,names[name] do t[#t+1] = 0 end
2734             t[#t+1] = value
2735             tableinsert(#ca == #cb and ca or cb, t)
2736         end
2737     end
2738     for _,t in ipairs{ca,cb} do
2739         for _,tt in ipairs(t) do
2740             for i=1,#names-#tt do tt[#tt+1] = 0 end
2741         end
2742     end
2743     if #names == 1 then
2744         colorspace = objref
2745     else

```

```

2746     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2747   end
2748 else
2749   local model = 0
2750   for _,t in ipairs{ca,cb} do
2751     for _,tt in ipairs(t) do
2752       model = model > #tt and model or #tt
2753     end
2754   end
2755   for _,t in ipairs{ca,cb} do
2756     for _,tt in ipairs(t) do
2757       if #tt < model then
2758         color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2759       end
2760     end
2761   end
2762   colorspace = model == 4 and "/DeviceCMYK"
2763               or model == 3 and "/DeviceRGB"
2764               or model == 1 and "/DeviceGray"
2765               or err"unknown color model"
2766 end
2767 local extend = prescript.sh_extend
2768 local mesh_matrix
2769 if sh_type == "linear" then
2770   local coordinates = format("%f %f %f %f",
2771     dx + sx*centera[1], dy + sy*centera[2],
2772     dx + sx*centerb[1], dy + sy*centerb[2])
2773   shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2774 elseif sh_type == "circular" then
2775   local factor = prescript.sh_factor or 1
2776   local radiusa = factor * prescript.sh_radius_a
2777   local radiusb = factor * prescript.sh_radius_b
2778   local coordinates = format("%f %f %f %f %f %f",
2779     dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2780     dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2781   shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2782 elseif sh_type == "coons" or sh_type == "tensor" then
2783   shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2784 elseif sh_type == "triangle" then
2785   shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2786 elseif sh_type == "lattice" then
2787   shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2788 else
2789   err"unknown shading type"
2790 end
2791
2792 local matrix = prescript.sh_matrix
2793 if matrix and matrix:find"%a" then
2794   local t = get_mp_matrix(matrix)

```

```

2795     matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2796 end
2797 if mesh_matrix and matrix then
2798     local a, b = mesh_matrix:explode(), matrix:explode()
2799     matrix = format("%f %f %f %f %f %f",
2800         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],
2801         a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)
2802 end
2803
2804 return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix
2805 end
2806 end
2807

```

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

```

2808 local function add_pattern_resources (key, val)
2809     if pdfmanagement then
2810         texsprint (
2811             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2812         )
2813     else
2814         local res = format("/%s %s", key, val)
2815         if is_defined(pdfetcs.pgfpattern) then
2816             texsprint ( "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" )
2817         else
2818             pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2819         end
2820     end
2821 end
2822 if pdfmode then
2823     function luamplib.dolatelua (on, os, matrix)
2824         local h, v = pdf.getpos()
2825         local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2826         matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2827             :gsub(decimals,rmzeros)
2828         pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2829         pdf.refobj(on)
2830     end
2831 else
2832     pdfetcs.shadingpatterns = { }
2833     pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2834     function luamplib.dolatelua (on, kind, xobj)
2835         local h, v
2836         local t = pdfetcs.shadingpatterns[on] or { }
2837         local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2838             or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2839         if shift then
2840             h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2841         else

```

```

2842     h, v = pdf.getpos()
2843     h = format("%f", h/factor) :gsub(decimals,rmzeros)
2844     v = format("%f", v/factor) :gsub(decimals,rmzeros)
2845 end
2846 if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2847     warn"Rerun to get correct shading pattern"
2848 end
2849 local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2850 local init = pdfetcs.shadingpatterninit_w
2851 if init then pdfetcs.shadingpatterninit_w = nil end
2852 local f = ioopen(name, init and "w" or "a")
2853 if f then
2854     f:write(("s s s\n"):format(on, h, v))
2855     f:close()
2856 else
2857     err"cannot write a file. check the cache dir path"
2858 end
2859 end
2860 end
2861 local function do_preobj_shading (object, prescript)
2862     if not prescript or not prescript.sh_operand_type then return end
2863     local on,_,matrix = do_preobj_SH(object, prescript)
2864     local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2865     matrix = matrix or "1 0 0 1 0 0"
2866     if not pdfmode and prescript.sh_in_xobj == "yes" then -- only in dvi mode
2867         on = update_pdfobjs(("<<s/Matrix[%s]>>"):format(os, matrix))
2868         goto skip_latelua
2869     end
2870     on = update_pdfobjs()
2871     if pdfmode then
2872         put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"]],[",matrix,"]])"}))
2873     else
2874         local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2875                 or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2876         local init = pdfetcs.shadingpatterninit_r
2877         if init then
2878             pdfetcs.shadingpatterninit_r = nil
2879             local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2880             local f = ioopen(name)
2881             if f then
2882                 for line in f:lines() do
2883                     local t = line:explode()
2884                     pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2885                 end
2886                 f:close()
2887             end
2888         end

```

This seems to be needed for proper functioning:

```

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

2889 local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2890 local mt = matrix:explode()
2891 matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2892 texsprintf( "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2893             format(" <<%s/Matrix[%s]>>}", os, matrix) )
2894 put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2895            xobj and ("%s',[[%s]]"):format(xobj[1], xobj[2]))
2896 end
2897 ::skip_latelua::
2898 local key, val = format("MPlibPt%s", on), format(pdfetcs.resfmt, on)
2899 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.

```

2900 local stroke = prescript.sh_stroking
2901 if stroke == "filldraw" or stroke == "both" then
2902   pdf_literalcode("/Pattern cs/%s scn /Pattern CS/%s SCN", key, key)
2903 elseif stroke == "draw" or stroke == "yes" then
2904   pdf_literalcode("/Pattern CS/%s SCN", key)
2905 else
2906   pdf_literalcode("/Pattern cs/%s scn", key)
2907 end

```

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

```

2908 prescript.sh_type = nil
2909 end
2910

```

Tiling Patterns

```

2911 pdfetcs.patterns = { }
2912 local function gather_resources (optres, ispattern)
2913   local t = { }
2914   if pdfmanagement then
2915     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2916       local mytoks
2917       run_tex_code ({
2918         "\\mplibtmptoks\\expanded{{" ,
2919         "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/" , v , "}" ,
2920         "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/" , v , "}}", "}" ,
2921       }, ccexplat)
2922       mytoks = texgettoks"mplibtmptoks"
2923       if not pdfmode then
2924         mytoks = mytoks:gsub("\\str_convert_pdfname:n%s*{(.-)}", "%1") -- why not expanded?
2925       end
2926       mytoks = mytoks and mytoks:gsub("^%s*(.)%s*$", "%1")

```

```

2927     if mytoks and mytoks ~= "" then
2928         t[#t+1] = ("%s<<%s>>"):format(v, mytoks)
2929     end
2930 end
2931 elseif is_defined(pdfetcs.pgfbextgs) then
2932     run_tex_code"\relax" -- flush tex.sprint queue
2933     if pdfmode then
2934         for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2935                             ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2936                             Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2937             local res = (get_macro(v) or ""):gsub("^%s*(.)%s*$", "%1")
2938             if res ~= "" then
2939                 t[#t+1] = ("%s<<%s>>"):format(k, res )
2940             end
2941         end
2942     else
2943         local abc = get_macro"pgfutil@abc" or ""
2944         for k,v in pairs { ExtGState = "@pgfbextgs",
2945                             ColorSpace = "@pgfcolorspaces",
2946                             Pattern = "@pgfpatterns", } do
2947             local tt = { }
2948             for vv in abc:gmatch( v .. "%s*(%b<>)" ) do
2949                 tt[#tt+1] = vv:match("^<<%s*(.)%s*>>$")
2950             end
2951             if #tt > 0 then
2952                 t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2953             end
2954         end
2955     end

```

We still have to deal with Shading resources.

```

2956     if luatexbase.callbacktypes.finish_pdffile then
2957         if pdfetcs.Shading_res then
2958             t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2959         end
2960     else
2961         local res = pdfetcs.getpageres()
2962         res = res and res:match"/Shading%s*%b<>"
2963         if res then
2964             t[#t+1] = res
2965         end
2966     end
2967 else
2968     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

```

2969     warn"transparent package is not fully functional without pdfmanagement code."
2970 end
2971 if luatexbase.callbacktypes.finish_pdffile then

```

```

2972     for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2973         local tt = pdfetcs[v.."_res"]
2974         if tt then
2975             t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2976         end
2977     end
2978 else
2979     local res = pdfetcs.getpageres()
2980     if res then
2981         t[#t+1] = res
2982     end
2983 end
2984 end
2985 local result = tableconcat(t)
2986 if optres ~= "" then
2987     for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2988         local res = optres:match("/"..v.."%"s*%b<>")
2989         if res then
2990             if result:find("/"..v) then
2991                 res = res:match("<<(.+)>>$")
2992                 result = result:gsub("/"..v.."%"s*<<," %1"..res, 1)
2993             else
2994                 result = result .. res
2995             end
2996         end
2997     end
2998 end
2999 return result
3000 end
3001 function luamplib.registerpattern ( boxid, name, opts )
3002     local box = texgetbox(boxid)
3003     local wd = format("%.3f",box.width/factor)
3004     local hd = format("%.3f",(box.height+box.depth)/factor)
3005     info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
3006     if opts.xstep == 0 then opts.xstep = nil end
3007     if opts.ystep == 0 then opts.ystep = nil end
3008     if opts.colored == nil then
3009         opts.colored = opts.coloured
3010         if opts.colored == nil then
3011             opts.colored = true
3012         end
3013     end
3014     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
3015     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
3016     if opts.matrix and opts.matrix:find"%a" then
3017         local t = get_mp_matrix(opts.matrix)
3018         opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
3019         opts.xshift = opts.xshift or format("%f",t[5])
3020         opts.yshift = opts.yshift or format("%f",t[6])

```

```

3021 end
3022 local attr = {
3023   "/Type/Pattern",
3024   "/PatternType 1",
3025   format("/PaintType %i", opts.colored and 1 or 2),
3026   "/TilingType 2",
3027   format("/XStep %s", opts.xstep or wd),
3028   format("/YStep %s", opts.ystep or hd),
3029   format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
3030 }
3031 local optres = opts.resources or ""
3032 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
3033 local patterns = pdfetcs.patterns
3034 if pdfmode then
3035   if opts.bbox then
3036     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3037   end
3038   attr = tableconcat(attr) :gsub(decimals,rmzeros)
3039   local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
3040   patterns[name] = { id = index, colored = opts.colored }
3041 else
3042   local cnt = #patterns + 1
3043   local objname = "@mplibpattern" .. cnt
3044   local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
3045   texsprint (
3046     "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
3047     "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
3048     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3049     "\\special{pdf:bcontent}",
3050     "\\special{pdf:bxobj ", objname, " ", metric, "}",
3051     "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
3052     "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
3053     "\\special{pdf:put @resources <<", optres, ">>}",
3054     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3055     "\\special{pdf:econtent}}")
3056   )
3057   patterns[cnt] = objname
3058   patterns[name] = { id = cnt, colored = opts.colored }
3059   patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
3060 end
3061 end
3062
3063 local do_preobj_PAT
3064 do
3065   local function pattern_colorspace (cs)
3066     local on, new = update_pdfobjs(format("/Pattern %s]", cs))
3067     if new then
3068       local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
3069       if pdfmanagement then

```

```

3070      texsprint (
3071          "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
3072      )
3073  else
3074      local res = format("/%s %s", key, val)
3075      if is_defined(pdfetcs.pgfcolorspace) then
3076          texsprint ( "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" )
3077      else
3078          pdfetcs.fallback_update_resources("ColorSpace",res,"@MPLibCS")
3079      end
3080  end
3081  end
3082  return on
3083  end
3084  local function uncoloredGS(color,key,fill)
3085      local cs
3086      if color:find" cs " then
3087          local name
3088          if fill then
3089              name, color = color:match"^/(.-) cs (.-) sc"
3090          else
3091              name, color = color:match" /(-) CS (-) SC"
3092          end
3093          if pdfmode then
3094              cs = format("%s 0 R", ltx.pdf.object_id( name ))
3095              if cs == "0 0 R" then -- assumes colorspace.sty
3096                  _, cs = get_spc_name_obj("/"..name)
3097              end
3098          else
3099              run_tex_code({ "\\mplibtmptoks\\expanded{\\pdf_object_ref:n{", name, "}}"} }, ccexplat)
3100              cs = texgettoks"mplibtmptoks"
3101          end
3102      else
3103          if fill then
3104              color = colorsplit(color)
3105          else
3106              color = color:match"%a+ (.-) %a+":explode()
3107          end
3108          cs = #color == 4 and "/DeviceCMYK" or #color == 3 and "/DeviceRGB" or "/DeviceGray"
3109          color = tableconcat(color," ")
3110      end
3111      return fill and format("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3112          or format("/MPLibCS%i CS %s /%s SCN", pattern_colorspace(cs), color, key)
3113  end
3114  function do_preobj_PAT(object, prescript)
3115      local name = prescript and prescript.mplibpattern
3116      if not name then return end
3117      local patterns = pdfetcs.patterns
3118      local patt = patterns[name]

```

```

3119 local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3120 local key = format("MPLibPt%s",index)
3121 local stroke, fillgs, strkgs = prescript.mplibpatternstroke
3122 if patt.colored then
3123     fillgs = format("/Pattern cs /%s scn", key)
3124     strkgs = stroke and format("/Pattern CS /%s SCN", key)
3125 else
3126     local color = prescript.mpliboverridecolor
3127     if not color then
3128         local t = object.color
3129         color = t and #t>0 and luamplib.colorconverter(t)
3130     end
3131     if not color then return end
3132     fillgs = uncoloredGS(color,key,true)
3133     strkgs = stroke and uncoloredGS(color,key,false)
3134 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.

```

3135 if stroke == "filldraw" or stroke == "both" then
3136     pdf_literalcode("%s %s", fillgs, strkgs)
3137 elseif stroke == "draw" or stroke == "yes" then
3138     pdf_literalcode(strkgs)
3139 else
3140     pdf_literalcode(fillgs)
3141 end
3142 if not patt.done then
3143     local val = pdfmode and format("%s 0 R",index) or patterns[index]
3144     add_pattern_resources(key,val)
3145 end
3146 patt.done = true
3147 end
3148 end
3149

```

Fading

```

3150 pdfetcs.fading = { }
3151 local function do_preobj_FADE (object, prescript)
3152     local fd_type = prescript and prescript.mplibfadetype
3153     local fd_stop = prescript and prescript.mplibfadestate
3154     if not fd_type then
3155         return fd_stop -- returns "stop" (if picture) or nil
3156     end
3157     local on, os, new
3158     if fd_type == "masking" then
3159         local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3160         on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
3161         local bc = prescript.mplibmaskingbgcolor
3162         bc = bc and bc:gsub(":", " ")
3163         bc = bc and ("BC[%s]"):format(bc):gsub(decimals,rmzeros) or ""

```

```

3164     os = format("<</SMask<</S/Luminosity/G %s%>>>>",
3165                 pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3166 else
3167     local bbox = prescript.mplibfadebbox:explode":""
3168     local vec = prescript.mplibfadevector; vec = vec and vec:explode":""
3169     if not vec then
3170         if fd_type == "linear" then
3171             vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3172         else
3173             local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3174             vec = {centerx, centery, centerx, centery} -- center for both circles
3175         end
3176     end
3177     local coords = { vec[1], vec[2], vec[3], vec[4] }
3178     if fd_type == "linear" then
3179         coords = format("%f %f %f %f", tableunpack(coords))
3180     elseif fd_type == "circular" then
3181         local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3182         local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":""
3183         tableinsert(coords, 3, radius[1])
3184         tableinsert(coords, radius[2])
3185         coords = format("%f %f %f %f %f %f", tableunpack(coords))
3186     else
3187         err("unknown fading method '%s'", fd_type)
3188     end
3189     fd_type = fd_type == "linear" and 2 or 3
3190     local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3191     local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":"" }
3192     local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":"" }
3193     if steps > 1 then
3194         fractions = { prescript.sh_fraction_1 or 0 }
3195         for i=2,steps do
3196             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3197             ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":""
3198             cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":""
3199         end
3200     end
3201
3202     local wd = prescript.mplibadjustbbox
3203     if not wd then
3204         local pen = mplib.pen_info(object)
3205         if pen and pen.width then
3206             wd = pen.width / 2 * math.sqrt(2)
3207         end
3208     end
3209     if wd then
3210         bbox = { bbox[1]-wd, bbox[2]-wd, bbox[3]+wd, bbox[4]+wd }
3211     end
3212

```

```

3213 local dx, dy = -bbox[1], -bbox[2]
3214 local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3215 matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3216 matrix[5] = matrix[5] + dx
3217 matrix[6] = matrix[6] + dy
3218 matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3219 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3220 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3221 on = update_pdfobjs(os)
3222 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3223 local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
3224 :gsub(decimals,rmzeros)
3225 os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3226 on = update_pdfobjs(os)
3227 local resources = format(pdfetcs.resfmt, on)
3228 on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
3229 local attr = tableconcat{
3230     "/Subtype/Form",
3231     "/BBox[" .. bbox .. "]",
3232     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
3233     "/Resources " .. resources,
3234     "/Group " .. format(pdfetcs.resfmt, on),
3235 } :gsub(decimals,rmzeros)
3236 on = update_pdfobjs(attr, streamtext)
3237 os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
3238 end
3239 on, new = update_pdfobjs(os)
3240 local key = add_extgs_resources(on,new)
3241 start_pdf_code()
3242 pdf_literalcode("/%s gs", key)
3243 if fd_stop then return "standalone" end
3244 return "start"
3245 end
3246

```

Transparency Group

```

3247 pdfetcs.tr_group = { shifts = { } }
3248 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3249 local function do_preobj_GRP (object, prescript)
3250     local grstate = prescript and prescript.gr_state
3251     if not grstate then return end
3252     local trgroup = pdfetcs.tr_group
3253     if grstate == "start" then
3254         trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3255         trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3256         trgroup.wrapped = false
3257         for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+) do
3258             trgroup[v] = true
3259         end

```

```

3260   trgroup.bbox = prescript.mplibgroupbbox:explode":"
3261   trgroup.adjustbbox = prescript.mplibadjustbbox
3262   if not trgroup.adjustbbox then
3263     trgroup.widths = { }
3264   end
3265   put2output[["\begingroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3266 elseif grstate == "stop" then
3267   local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3268
3269   local wd = trgroup.adjustbbox
3270   if not wd then
3271     if #trgroup.widths > 0 then
3272       wd = math.max(tableunpack(trgroup.widths))
3273       if wd > 0 then
3274         wd = wd / 2 * math.sqrt(2)
3275       end
3276     end
3277     trgroup.widths = nil
3278   end
3279   if wd then
3280     llx,lly,urx,ury = llx-wd, lly-wd, urx+wd, ury+wd
3281   end
3282
3283   put2output(tableconcat{
3284     "\\egroup",
3285     format("\\wd\\mplibscratchbox %fbp", urx-llx),
3286     format("\\ht\\mplibscratchbox %fbp", ury-lly),
3287     "\\dp\\mplibscratchbox 0pt",
3288   })
3289   local grattr
3290   if trgroup.off then
3291     grattr = ""
3292   else
3293     local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>>",
3294                                     trgroup.isolated, trgroup.knockout))
3295     grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3296   end
3297   local res = gather_resources("")
3298   local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3299   if pdfmode then
3300     if trgroup.wrapped then
3301       put2output(tableconcat{
3302         "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3303         "/BBox[" .. bbox .. "]} resources{" .. res .. "}\\mplibscratchbox",
3304         "\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}",
3305       })
3306     end
3307     put2output(tableconcat{
3308       "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",

```

```

3309     "/BBox[" , bbox , "]" , grattr , "}" resources{ , res , "}" \mplibscratchbox" ,
3310     "\luamplibtagasgroupput{ , trgroup.name , "}" ,
3311     [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
3312     [[\wd\mplibscratchbox \opt\ht\mplibscratchbox \opt\dp\mplibscratchbox \opt]],
3313     [[\box\mplibscratchbox]],
3314     "}" \endgroup" ,
3315     "\expandafter\edef\csname luamplib.group." , trgroup.name , "\endcsname{ ,
3316     "\setbox\mplibscratchbox\hbox{\hskip" , -llx , "bp\raise" , -lly , "bp\hbox{ ,
3317     "\useboxresource \the\lastsavedboxresourceindex" ,
3318     "}}\wd\mplibscratchbox" , urx-llx , "bp\ht\mplibscratchbox" , ury-lly , "bp" ,
3319     "\box\mplibscratchbox}" ,
3320     })
3321   else
3322     if trgroup.wrapped then
3323       trgroup.cnt = (trgroup.cnt or 0) + 1
3324       local objname = format("@mplibtrgr%s" , trgroup.cnt)
3325       put2output(tableconcat{
3326         "\special{pdf:bxobj " , objname , " bbox " , bbox , "}" ,
3327         "\unhbox\mplibscratchbox" ,
3328         "\special{pdf:put @resources <<" , res , ">>}" ,
3329         "\special{pdf:exobj}" ,
3330         "\setbox\mplibscratchbox\hbox{\special{pdf:uxobj " , objname , "}" ,
3331       })
3332     end
3333     trgroup.cnt = (trgroup.cnt or 0) + 1
3334     local objname = format("@mplibtrgr%s" , trgroup.cnt)
3335     put2output(tableconcat{
3336       "\special{pdf:bxobj " , objname , " bbox " , bbox , "}" ,
3337       "\unhbox\mplibscratchbox" ,
3338       "\special{pdf:put @resources <<" , res , ">>}" ,
3339       "\special{pdf:exobj <<" , grattr , ">>}" ,
3340       "\luamplibtagasgroupput{ , trgroup.name , "}" ,
3341       "\special{pdf:uxobj " , objname , "}" ,
3342       "}" \endgroup" ,
3343     })
3344     token.set_macro("luamplib.group." .. trgroup.name , tableconcat{
3345       "\setbox\mplibscratchbox\hbox{\hskip" , -llx , "bp\raise" , -lly , "bp\hbox{ ,
3346       "\special{pdf:uxobj " , objname , "}" ,
3347       "}}\wd\mplibscratchbox" , urx-llx , "bp\ht\mplibscratchbox" , ury-lly , "bp" ,
3348       "\box\mplibscratchbox" ,
3349     } , "global")
3350   end
3351   trgroup.shifts[trgroup.name] = { llx , lly }
3352 end
3353 return grstate
3354 end
3355 function luamplib.registergroup (boxid , name , opts)
3356   if opts.asgroup and opts.asgroup:find"wrapped" then
3357     luamplib.registergroup(boxid , name , {bbox=opts.bbox , resources=opts.resources})

```

```

3358     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}"}
3359     opts.asgroup = opts.asgroup:gsub("wrapped", "")
3360 end
3361 local box = texgetbox(boxid)
3362 local wd, ht, dp = node.getwhd(box)
3363 local is_mask = opts.asgroup and opts.asgroup:find"masking"
3364 local res = opts.resources or ""
3365 res = gather_resources(res)
3366 local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3367 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
3368 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
3369 if opts.matrix and opts.matrix:find"%a" then
3370     local t = get_mp_matrix(opts.matrix)
3371     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
3372 end
3373 local grtype = 3
3374 if opts.bbox then
3375     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3376     grtype = 2
3377 end
3378 local mpllx, mplly = get_macro'MPlLx', get_macro'MPlLy'
3379 if is_mask then
3380     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3381     t[5], t[6] = t[5]+mpllx, t[6]+mplly
3382     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
3383     mpllx, mplly = 0, 0
3384 end
3385 if opts.matrix then
3386     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3387     grtype = opts.bbox and 4 or 1
3388 end
3389 if opts.asgroup and not opts.asgroup:find"off" then
3390     local t = { isolated = false, knockout = false, masking = false }
3391     for _, v in ipairs(opts.asgroup:gsub("%s", ""):explode",+") do t[v] = true end
3392     local on
3393     if t.masking then
3394         on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorsname or "/DeviceGray"))
3395     else
3396         local cs = opts.colorsname and ("/CS%s"):format(opts.colorsname) or ""
3397         on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3398     end
3399     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3400 end
3401 local trgroup = pdfetcs.tr_group
3402 trgroup.shifts[name] = { mpllx, mplly }
3403 local whd
3404 if pdfmode then
3405     attr = tableconcat(attr, :gsub(decimals, rmzeros))
3406     local index = tex.saveboxresource(boxid, attr, res, true, grtype)

```

```

3407 token.set_macro("luamplib.group"..name, tableconcat{
3408     "\\useboxresource ", index,
3409 }, "global")
3410 whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3411 else
3412     trgroup.cnt = (trgroup.cnt or 0) + 1
3413     local objname = format("@mplibtrgr%s", trgroup.cnt)
3414     texsprint (
3415         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3416         "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3417         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3418         "\\special{pdf:bcontent}",
3419         "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3420         "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3421         "\\special{pdf:put @resources <<", res, ">>}",
3422         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3423         "\\special{pdf:econtent}}")
3424     )
3425     token.set_macro("luamplib.group"..name, tableconcat{
3426         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3427         "\\wd\\mplibscratchbox ", wd, "sp",
3428         "\\ht\\mplibscratchbox ", ht, "sp",
3429         "\\dp\\mplibscratchbox ", dp, "sp",
3430         "\\box\\mplibscratchbox",
3431     }, "global")
3432     whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3433 end
3434 info("w/h/d of group '%s': %s", name, whd)
3435 end
3436

```

luamplib.convert: flushing figures

```

3437 do
3438     local function stop_special_effects(fade,opaq)
3439         if opaq then -- opacity
3440             pdf_literalcode(opaq)
3441         end
3442         if fade then -- fading
3443             stop_pdf_code()
3444         end
3445     end
3446

```

For parsing prescript materials.

```

3447 local function script2table(s)
3448     local t = {}
3449     for _,i in ipairs(s:explode("\\13+")) do
3450         local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3451         if k and v and k ~= "" and not t[k] then
3452             t[k] = v

```

```

3453     end
3454     end
3455     return t
3456 end
3457

```

For path shading with transformed pen

```

3458 local function invert_matrix (t)
3459     local a, b, c, d, x, y = t[1], t[2], t[3], t[4], t[5], t[6]
3460     local det = a*d - b*c
3461     if det == 0 then err"transformation is not invertible!" end
3462     return format("%f %f %f %f %f %f cm ",
3463         d/det, 0-b/det, 0-c/det, a/det, (d*x-b*y)/det, (a*y-c*x)/det)
3464 end
3465

```

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

```

3466 local function pdf_textfigure(font,size,text,width,height,depth)
3467     text = text:gsub(".",function(c)
3468         return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
3469     end)
3470     put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
3471 end
3472
3473 local bend_tolerance = 131/65536
3474
3475 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3476
3477 local function pen_characteristics(object)
3478     local t = mplib.pen_info(object)
3479     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3480     divider = sx*sy - rx*ry
3481     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3482 end
3483
3484 local function concat(px, py) -- no tx, ty here
3485     return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
3486 end
3487
3488 local function curved(ith,pth)
3489     local d = pth.left_x - ith.right_x
3490     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3491         abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3492         d = pth.left_y - ith.right_y
3493         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3494             abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3495             return false
3496         end
3497     end

```

```

3498     return true
3499 end
3500
3501 local function flushnormalpath(path,open)
3502     local pth, ith
3503     for i=1,#path do
3504         pth = path[i]
3505         if not ith then
3506             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3507         elseif curved(ith,pth) then
3508             pdf_literalcode("%f %f %f %f %f %f c",
3509                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3510         else
3511             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3512         end
3513         ith = pth
3514     end
3515     if not open then
3516         local one = path[1]
3517         if curved(pth,one) then
3518             pdf_literalcode("%f %f %f %f %f %f c",
3519                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3520         else
3521             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3522         end
3523     elseif #path == 1 then -- special case .. draw point
3524         local one = path[1]
3525         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3526     end
3527 end
3528
3529 local function flushconcatpath(path,open)
3530     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3531     local pth, ith
3532     for i=1,#path do
3533         pth = path[i]
3534         if not ith then
3535             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3536         elseif curved(ith,pth) then
3537             local a, b = concat(ith.right_x,ith.right_y)
3538             local c, d = concat(pth.left_x,pth.left_y)
3539             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3540         else
3541             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3542         end
3543         ith = pth
3544     end
3545     if not open then
3546         local one = path[1]

```

```

3547     if curved(pth,one) then
3548         local a, b = concat(pth.right_x,pth.right_y)
3549         local c, d = concat(one.left_x,one.left_y)
3550         pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3551     else
3552         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3553     end
3554 elseif #path == 1 then -- special case .. draw point
3555     local one = path[1]
3556     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3557 end
3558 end
3559

```

Finally, flush figures by inserting PDF literals.

```

3560 local function flush (result,flusher)
3561     if result then
3562         local figures = result.fig
3563         if figures then
3564             for f=1, #figures do
3565                 info("flushing figure %s",f)
3566                 local figure = figures[f]
3567                 local objects = figure:objects()
3568                 local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3569                 local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3570                 local bbox = figure:boundingbox()
3571                 local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3572                 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()

```

```

3573     else

```

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

```

3574         if tex_code_pre_mplib[f] then
3575             put2output(tex_code_pre_mplib[f])
3576         end
3577         pdf_startfigure(fignum,llx,lly,urx,ury)
3578         start_pdf_code()
3579         if objects then
3580             local savedpath = nil
3581             local savedhtap = nil
3582             for o=1,#objects do
3583                 local object      = objects[o]
3584                 local objecttype  = object.type

```

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

```

3585         local prescript      = object.prescript
3586         prescript = prescript and script2table(prescript) -- prescript is now a table
3587         do_preobj_CR(object,prescript) -- color
3588         local fading_ = do_preobj_FADE(object,prescript) -- fading
3589         local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3590         do_preobj_PAT(object,prescript) -- tiling pattern
3591         do_preobj_shading(object,prescript) -- shading pattern
3592         local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3593         if prescript and prescript.mplibtexboxid then
3594             put_tex_boxes(object,prescript)
3595         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3596         elseif objecttype == "start_clip" then
3597             local evenodd = not object.istext and object.postscript == "evenodd"
3598             start_pdf_code()
3599             flushnormalpath(object.path,false)
3600             pdf_literalcode(evenodd and "W* n" or "W n")
3601         elseif objecttype == "stop_clip" then
3602             stop_pdf_code()
3603             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3604         elseif objecttype == "special" then

```

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

```

3605         if prescript and prescript.postmplibverbtx then
3606             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
3607         end
3608         elseif objecttype == "text" then
3609             local ot = object.transform -- 3,4,5,6,1,2
3610             start_pdf_code()
3611             pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3612             pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3613             stop_pdf_code()
3614         elseif not trgroup and fading_ ~= "stop" then
3615             local evenodd, collect, both = false, false, false
3616             local postscript = object.postscript
3617             if not object.istext then
3618                 if postscript == "evenodd" then
3619                     evenodd = true
3620                 elseif postscript == "collect" then
3621                     collect = true
3622                 elseif postscript == "both" then
3623                     both = true
3624                 elseif postscript == "eoboth" then
3625                     evenodd = true
3626                     both = true
3627                 end
3628             end
3629             if collect then
3630                 if not savedpath then

```

```

3631         savedpath = { object.path or false }
3632         savedhtap = { object.htap or false }
3633     else
3634         savedpath[#savedpath+1] = object.path or false
3635         savedhtap[#savedhtap+1] = object.htap or false
3636     end
3637 else

```

Removed from ConTeXt general: color stuff.

```

3638         local ml = object.miterlimit
3639         if ml and ml ~= miterlimit then
3640             miterlimit = ml
3641             pdf_literalcode("%f M",ml)
3642         end
3643         local lj = object.linejoin
3644         if lj and lj ~= linejoin then
3645             linejoin = lj
3646             pdf_literalcode("%i j",lj)
3647         end
3648         local lc = object.linecap
3649         if lc and lc ~= linecap then
3650             linecap = lc
3651             pdf_literalcode("%i J",lc)
3652         end
3653         local dl = object.dash
3654         if dl then
3655             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3656             if d ~= dashed then
3657                 dashed = d
3658                 pdf_literalcode(dashed)
3659             end
3660         elseif dashed then
3661             pdf_literalcode("[ ] 0 d")
3662             dashed = false
3663         end
3664         local path = object.path
3665         local transformed, penwidth = false, 1
3666         local open = path and path[1].left_type and path[#path].right_type
3667         local pen = object.pen
3668         if pen then
3669             if pen.type == 'elliptical' then
3670                 transformed, penwidth = pen_characteristics(object) -- boolean, value

```

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

```

3671             if penwidth and pdfetcs.tr_group.widths then
3672                 tableinsert(pdfetcs.tr_group.widths, penwidth)
3673             end
3674             pdf_literalcode("%f w",penwidth)
3675             if objecttype == 'fill' then
3676                 objecttype = 'both'

```

```

3677         end
3678     else -- calculated by mplib itself
3679         objecttype = 'fill'
3680     end
3681 end

```

Added : shading

```

3682 local shade_no, shade_stroke, shade_cm, shade_im = do_preobj_SH(object,prescript) -- shading
3683 if shade_no then
3684     objecttype = "shade"
3685     shade_im = transformed and invert_matrix{sx, rx, ry, sy, tx ,ty} or ""
3686     shade_cm = shade_cm and shade_cm.." cm " or ""
3687 end
3688 if transformed then
3689     start_pdf_code()
3690 end
3691 if path then
3692     if savedpath then
3693         for i=1,#savedpath do
3694             local path = savedpath[i]
3695             if transformed then
3696                 flushconcatpath(path,open)
3697             else
3698                 flushnormalpath(path,open)
3699             end
3700         end
3701         savedpath = nil
3702     end
3703     if transformed then
3704         flushconcatpath(path,open)
3705     else
3706         flushnormalpath(path,open)
3707     end
3708     if objecttype == "fill" then
3709         pdf_literalcode(evenodd and "h f*" or "h f")
3710     elseif objecttype == "outline" then
3711         if both then
3712             pdf_literalcode(evenodd and "h B*" or "h B")
3713         else
3714             pdf_literalcode(open and "S" or "h S")
3715         end
3716     elseif objecttype == "both" then
3717         pdf_literalcode(evenodd and "h B*" or "h B")
3718     elseif objecttype == "shade" then
3719         pdf_literalcode("q W%s %s %s%s/MPlibSh%s sh Q",
3720             evenodd and "*" or "",
3721             shade_stroke and "s" or "n",
3722             shade_im, shade_cm, shade_no)
3723     end

```

```

3724         end
3725         if transformed then
3726             stop_pdf_code()
3727         end
3728         local path = object.htap
htap objects are generated by, say, filldraw with pensquare.
3729         if path then
3730             if transformed then
3731                 start_pdf_code()
3732             end
3733             if savedhtap then
3734                 for i=1,#savedhtap do
3735                     local path = savedhtap[i]
3736                     if transformed then
3737                         flushconcatpath(path,open)
3738                     else
3739                         flushnormalpath(path,open)
3740                     end
3741                 end
3742                 savedhtap = nil
3743                 evenodd = true
3744             end
3745             if transformed then
3746                 flushconcatpath(path,open)
3747             else
3748                 flushnormalpath(path,open)
3749             end
3750             if objecttype == "fill" then
3751                 pdf_literalcode(evenodd and "h f*" or "h f")
3752             elseif objecttype == "outline" then
3753                 pdf_literalcode(open and "S" or "h S")
3754             elseif objecttype == "both" then
3755                 pdf_literalcode(evenodd and "h B*" or "h B")
3756             elseif objecttype == "shade" then
3757                 pdf_literalcode("q W%s %s %s%s/MPlibSh%s sh Q",
3758                     evenodd and "*" or "",
3759                     shade_stroke and "s" or "n",
3760                     shade_im, shade_cm, shade_no)
3761             end
3762             if transformed then
3763                 stop_pdf_code()
3764             end
3765         end
3766     end
3767 end

```

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

Note that fading itself is just a ExtGS, such that opacity should have effect on individual objects just like color. (color is always reset in front of each graphic object.) The order therefore

should be: fading (q) — opacity — graphic object — stop opacity — ... — stop fading (Q).

To be frank, a pattern (especially shading) is also a ExtGS. However, as there are so many optional macros and, above all, they are *reset by the color* of each object, we decided them to affect all the following objects which have no color.

```

3768         if fading_ == "start" then
3769             stop_special_effects(false, tr_opaq)
3770             pdfetcs.fading.specialeffects = {fading_, false}
3771         elseif trgroup == "start" then
3772             pdfetcs.tr_group.specialeffects = {fading_, tr_opaq}
3773         elseif fading_ == "stop" then
3774             local se = pdfetcs.fading.specialeffects
3775             if se then stop_special_effects(se[1], se[2]) end
3776         elseif trgroup == "stop" then
3777             local se = pdfetcs.tr_group.specialeffects
3778             if se then stop_special_effects(se[1], se[2]) end
3779         else
3780             stop_special_effects(fading_, tr_opaq)
3781         end
3782         if fading_ or trgroup then -- extgs resetted
3783             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3784         end
3785     end
3786 end
3787 stop_pdf_code()
3788 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtex code.

```

3789     for _,v in ipairs(figcontents) do
3790         if type(v) == "table" then
3791             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3792         else
3793             texsprint(v)
3794         end
3795     end
3796     if #figcontents.post > 0 then texsprint(figcontents.post) end
3797     figcontents = { post = { } }
3798 end
3799 end
3800 end
3801 end
3802 end
3803
3804 function luamplib.convert (result, flusher)
3805     flush(result, flusher)
3806     return true -- done
3807 end
3808 end
3809
3810 function luamplib.colorconverter (cr)

```

```

3811 local n = #cr
3812 if n == 4 then
3813     local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3814     return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3815 elseif n == 3 then
3816     local r, g, b = cr[1], cr[2], cr[3]
3817     return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3818 else
3819     local s = cr[1]
3820     return format("%.3f g %.3f G",s,s), "0 g 0 G"
3821 end
3822 end
3823

```

The code for externalize functionality

```

3824 do
3825     local extname, prefix, pdfname, luaname, figtab, prevfigtab, bigname, bigtime
3826     local majorV, minorV = pdf.getmajorversion(), pdf.getminorversion()
3827     local function is_shell_esc()
3828         if status.shell_escape == 1 then return true end
3829         luatexbase.module_error("luamplib",
3830             "--shell-escape is needed for 'externalize' feature.\n\z
3831             Rerun with --shell-escape option.")
3832     end
3833     local function args_normalized ()
3834         local args = { }
3835         for i = exeidx, #arg do
3836             args[#args+1] = arg[i]
3837         end
3838         tableinsert(args, 2, "--halt-on-error")
3839         tableinsert(args, 3, "--interaction=nonstopmode")
3840         return args
3841     end
3842
3843     function externalize.setup ()
3844         extname = externalize.MY_NAME or format("%s-mplib-figure",tex.jobname)
3845         externalize.running = true
3846         local dir = status.output_directory or "."
3847         prefix = format("%s/%s",dir,extname)
3848         luaname = format("%ss.lua",prefix)
3849         pdfname = format("%s/%s.pdf",dir,tex.jobname)
3850         bigname = format("%ss.pdf",prefix)
3851         bigtime = lfs.modification(bigname) or 0
3852
3853         if externalize.MY_NAME then
3854             local dir = prefix:match"(.)[/\\]"
3855             if not lfs.isdir(dir) then
3856                 luatexbase.module_error("luamplib", dir.." directory does not exist!")
3857             end
3858         end
3859     end
3860 end

```

```

3858     end
3859
3860     local extver = format("20261002.%s%s", majorV, minorV)
3861     figtab = { version = extver }
3862     if lfs.isfile(luaname) then
3863         prevfigtab = require(luaname)
3864     end
3865     if not prevfigtab or prevfigtab.version ~= extver then
3866         bigtime = 0
3867         prevfigtab = { }
3868         externalize.activate()
3869     end
3870
3871     luatexbase.add_to_callback("finish_pdffile", function()
3872         local mandatory = figtab.mandatory
3873         if mandatory then
3874             local depends = externalize.depends
3875             for i = 1, #figtab do
3876                 if mandatory[i] then
3877                     for _,v in ipairs(depends[i]) do
3878                         mandatory[v] = true
3879                     end
3880                 end
3881             end
3882         end
3883         if not figtab.found then
3884             table.tofile(luaname, figtab, "return")
3885             if not mandatory or not is_shell_esc() then return end
3886             luatexbase.add_to_callback("wrapup_run", function()
3887                 os.exec(args_normalized())
3888             end, "luamplib_externalize")
3889         end
3890     end,
3891     "luamplib_externalize")
3892 end
3893 function externalize.activate ()
3894     figtab.active = true
3895     texsprint"\AddToHook{shipout/before}\z
3896     {\directlua{luamplib.externalize.shipout(tex.getbox(\the\ShipoutBox))}}"
3897 end
3898 function externalize.latelua(wd,ht,dp,mgn)
3899     local llx, lly = pdf.getpos()
3900     local urx, ury = (llx+wd+mgn)/factor, (lly+ht+mgn)/factor
3901     llx, lly = (llx-mgn)/factor, (lly-dp-mgn)/factor
3902     local cropbox = ("%f %f %f %f"):format(llx, lly, urx, ury)
3903     pdf.setpageattributes(cropbox)
3904 end
3905 function externalize.shipout (head)
3906     local attr = luatexbase.attributes.luamplibexternalizeattr

```

```

3907 local getnext, has_attribute = node.getnext, node.has_attribute
3908 local curr = head
3909 while curr do
3910     if curr.head then
3911         local count = attr and has_attribute(curr,attr)
3912         if count then
3913             local page = (externalize.page or tex.count.ReadonlyShipoutCounter) + 1
3914             externalize.page = page
3915
3916             local fig = figtab[count]
3917             local wd, ht, dp = curr.width, curr.height, curr.depth
3918
3919             local latelua = node.new("whatsit", "late_lua")
3920             latelua.token = format("luamplib.externalize.latelua(%s,%s,%s,%s)", wd, ht, dp, fig.margin or 0)
3921             latelua.name = "luamplib.externalize.latelua"
3922             curr.head = node.insert_before(curr.head, curr.head, latelua)
3923
3924             fig.pages = fig.pages or { }
3925             tableinsert(fig.pages, page)
3926             fig.metric = fig.metric or { }
3927             tableinsert(fig.metric, { wd, ht, dp })
3928
3929             tex.setbox("mplibscratchbox", node.copy(curr))
3930             tex.shipout"mplibscratchbox"
3931
3932             figtab.found = true
3933         else
3934             externalize.shipout(curr.head)
3935         end
3936     elseif curr.leader and curr.leader.head then
3937         externalize.shipout(curr.leader.head)
3938     end
3939     curr = getnext(curr)
3940 end
3941
3942 if figtab.found then
3943     if not luatexbase.in_callback("wrapup_run", "luamplib_externalize") then
3944         if is_shell_esc() then
3945             externalize.wrapup_run()
3946         else
3947             luatexbase.add_to_callback("wrapup_run", function() end, "luamplib_externalize")
3948         end
3949     end
3950     texsprint"\DiscardShipoutBox"
3951 end
3952 end
3953 function externalize.wrapup_run ()
3954     luatexbase.add_to_callback("wrapup_run", function()
3955         table.tofile(luaname, figtab, "return")

```

```

3956
3957     if bigtime == 0 then
3958         file.copy(pdfname, bigname)
3959     else
3960         for i,v in ipairs(figtab) do
3961             if v.pages then
3962                 local mgn = v.margin and v.margin*2 or 0
3963                 for ii,vv in ipairs(v.pages) do
3964                     local wd, ht, dp = tableunpack(v.metric[ii])
3965                     local status = os.spawn{
3966                         format('%s/luatex', os.selfdir),
3967                         format('--jobname=%s-%s-%s', extname, i, ii),
3968                         '--halt-on-error',
3969                         '--interaction=nonstopmode',
3970                         format('\pagewidth=%ssp\pageheight=%ssp', wd+mgn, ht+dp+mgn),
3971                         format('\pdfvariable majorversion %s\pdfvariable minorversion %s',majorV,minorV),
3972                         '\pdfvariable horigin 0pt\pdfvariable vorigin 0pt\topskip=0pt\nopagenumbers',
3973                         format('\directlua{img.write{filename=[[%s]],page=%s,pagebox="crop"}}',pdfname,vv),
3974                         '\bye',
3975                     }
3976                     assert(status == 0, format("failed to generate external image No. %s!",i))
3977                     os.remove(format("%s-%s-%s.log", prefix, i, ii))
3978                 end
3979             end
3980         end
3981     end
3982
3983     os.exec(args_normalized())
3984 end,
3985 "luamplib_externalize")
3986 end
3987 function externalize.figure(data)
3988     local count = tex.count.luamplibexternalizecount + 1
3989     tex.setcount("global", "luamplibexternalizecount", count)
3990
3991     local prevfig = prevfigtab[count]
3992     local pdfpages, whd
3993     if prevfig then
3994         pdfpages = prevfig.pdfpages
3995             or prevfig.pages
3996             or prevfig.changed and { }
3997         whd = prevfig.metric or prevfig.whd
3998     end
3999     local depend = get_macro"luamplibexternalizedependson"
4000
4001     local margin = get_macro"luamplibexternalizemargin"
4002         margin = margin and tex.sp(margin)
4003
4004     figtab[count] = {

```

```

4005     data = data,
4006     depend = depend,
4007     margin = margin,
4008     pdfpages = pdfpages,
4009     whd = whd,
4010 }
4011
4012 if get_macro"luamplibexternalizedothis" == "false" then return end
4013
4014 local match = prevfig and prevfig.data == data
4015             and prevfig.depend == depend
4016             and prevfig.margin == margin
4017
4018 if prevfigtab.mandatory and prevfigtab.mandatory[count] then
4019     goto do_this_fig
4020 end
4021 if depend then
4022     local deps = depend:explode","
4023     for i,v in ipairs(deps) do
4024         local n = tonumber(v)
4025         if n < 0 then
4026             n = count + n
4027         end
4028         deps[i] = n
4029     end
4030
4031     local depends = externalize.depends or { }
4032     externalize.depends = depends
4033     depends[count] = deps
4034     for _,v in ipairs(deps) do
4035         depends[v] = depends[v] or { }
4036         tableinsert(depends[v], count)
4037     end
4038
4039     local done = 0
4040     for _,v in ipairs(deps) do
4041         if figtab[v].changed then
4042             done = done + 1
4043         end
4044     end
4045
4046     if done == #deps then -- all changed: OK
4047         goto do_this_fig
4048     elseif done == 0 and match then -- none changed and match: OK
4049     else
4050         match = true -- avoid error
4051         figtab.mandatory = figtab.mandatory or { }
4052         for _,v in ipairs(deps) do
4053             figtab.mandatory[v] = true

```

```

4054     end
4055   end
4056 end
4057
4058 if match then
4059   if not pdfpages then goto do_this_fig end
4060   for i,v in ipairs(pdfpages) do
4061     local spec
4062     local small = format("%s-%s-%s.pdf", prefix, count, i)
4063     local smalltime = lfs.modification(small) or 0
4064     if smalltime > bigtime then
4065       spec = format("filename=[[%s]]", small)
4066     elseif bigtime > 0 then
4067       spec = format("filename=[[%s]],page=%s,pagebox=[[crop]],keepopen=true", bigname, v)
4068     end
4069     local wd, ht, dp = tableunpack(whd[i])
4070     if not spec then goto do_this_fig end
4071     texsprint(ccexplat,
4072       "\\prependtombibbox\\hbox dir TLT\\bgroup",
4073       "\\tag_socket_use:nn{luamplib/figure/begin}\\l__luamplib_tag_alt_dflt_tl",
4074       "\\setbox\\mplibscratchbox\\vbox to", ht+dp, "sp{\\vss\\hbox to", wd, "sp{\\hss",
4075       "\\directlua{img.write{", spec, "}}\\hss}\\vss}",
4076       "\\dp\\mplibscratchbox=", dp, "sp",
4077       "\\ht\\mplibscratchbox=\\dimexpr\\ht\\mplibscratchbox-\\dp\\mplibscratchbox\\relax",
4078       "\\tag_socket_use:nnn{luamplib/figure/end}\\mplibscratchbox}\\box\\mplibscratchbox}",
4079       "\\egroup")
4080   end
4081   return true
4082 end
4083
4084 ::do_this_fig::
4085 if not figtab.active then
4086   externalize.activate()
4087 end
4088 figtab[count].changed = true
4089 texsprint "\\luamplibexternalizethisfigure"
4090 end
4091 local index = luatexbase.new_luafunction "luamplib_externalize"
4092 lua.get_functions_table()[index] = function()
4093   local data = token.scan_argument()
4094   if externalize.running and externalize.figure(data) then
4095     tex.setcount("l_tmpa_int", 1)
4096   else
4097     tex.setcount("l_tmpa_int", 0)
4098   end
4099 end
4100 token.set_lua("luamplib@externalized", index, "global")
4101 end

```

2.2 T_EX package

First we need to load some packages.

```
4102 \ifcsname ProvidesPackage\endcsname
```

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

```
4103 \NeedsTeXFormat{LaTeX2e}
4104 \ProvidesPackage{luamplib}
4105 [2026/10/02 v2.44.1 mplib package for LuaTeX]
4106 \fi
4107 \ifdefined\newluafunction\else
4108 \input ltluatex
4109 \fi
```

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

```
4110 \ifnum\outputmode=0
4111 \ifdefined\AddToHookNext
4112 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
4113 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
4114 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
4115 \else
4116 \input atbegshi.sty
4117 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
4118 \let\luamplibatfirstshipout\AtBeginShipoutFirst
4119 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
4120 \fi
4121 \fi
```

Loading of lua code.

```
4122 \directlua{require("luamplib")}
```

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

```
4123 \ifx\pdfoutput\undefined
4124 \let\pdfoutput\outputmode
4125 \fi
4126 \ifx\pdfliteral\undefined
4127 \protected\def\pdfliteral{\pdfextension literal}
4128 \fi
```

Set the format for METAPOST.

```
4129 \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.

```
4130 \ifnum\pdfoutput>0
4131 \let\mplibtoPDF\pdfliteral
4132 \else
```

```

4133 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
4134 \ifcsname PackageInfo\endcsname
4135   \PackageInfo{luamplib}{only dvipdfmx is supported currently}
4136 \else
4137   \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
4138 \fi
4139 \fi

```

To make mplibcode typeset always in horizontal mode.

```

4140 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
4141 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
4142 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

4143 \def\mplibsetupcatcodes{%
4144   %catcode`\{=12 %catcode`\}=12
4145   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
4146   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
4147 }

```

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

```

4148 \ifnum\outputmode>0 %
4149   \def\mplibputtextbox#1#2#3#4{%
4150     \pdfextension save\relax
4151     \vbox to 0pt{\vss
4152       \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
4153       \kern#3bp}%
4154     \pdfextension restore\relax}
4155 \else
4156   \def\mplibputtextbox#1#2#3#4{%
4157     \special{pdf:btrans matrix #4 #2 #3}%
4158     \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
4159     \special{pdf:etrans}}
4160 \fi

```

use Transparency Group

```

4161 \protected\def\usemplibgroup#1#2{\usemplibgroupmain}
4162 \def\usemplibgroupmain#1{%
4163   \prependtomplibbox\hbox dir TLT\bgroup
4164   \csname luamplib.group.#1\endcsname
4165   \egroup
4166 }
4167 \protected\def\mplibgroup#1{%
4168   \begingroup
4169   \def\MPllx{0}\def\MPlly{0}%
4170   \def\mplibgroupname{#1}%
4171   \mplibgroupgetnexttok
4172 }
4173 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
4174 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok= }

```

```

4175 \def\mplibgroupbranch{%
4176   \ifx [\nexttok
4177     \expandafter\mplibgroupopts
4178   \else
4179     \ifx\mplibsptoken\nexttok
4180       \expandafter\expandafter\expandafter\mplibgroupskipspace
4181     \else
4182       \let\mplibgroupoptions\empty
4183       \expandafter\expandafter\expandafter\mplibgroupmain
4184     \fi
4185   \fi
4186 }
4187 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
4188 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
4189 \protected\def\endmplibgroup{\egroup
4190   \directlua{ luamplib.registergroup(
4191     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
4192   )}%
4193   \endgroup
4194 }

```

Patterns

```

4195 {\def\:{\global\let\mplibsptoken= } \: }
4196 \protected\def\mppattern#1{%
4197   \begingroup
4198   \def\MPllx{0}\def\MPlly{0}%
4199   \def\mplibpatternname{#1}%
4200   \mplibpatterngetnexttok
4201 }
4202 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
4203 \def\mplibpatternskipspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
4204 \def\mplibpatternbranch{%
4205   \ifx [\nexttok
4206     \expandafter\mplibpatternopts
4207   \else
4208     \ifx\mplibsptoken\nexttok
4209       \expandafter\expandafter\expandafter\mplibpatternskipspace
4210     \else
4211       \let\mplibpatternoptions\empty
4212       \expandafter\expandafter\expandafter\mplibpatternmain
4213     \fi
4214   \fi
4215 }
4216 \def\mplibpatternopts[#1]{%
4217   \def\mplibpatternoptions{#1}%
4218   \mplibpatternmain
4219 }
4220 \def\mplibpatternmain{%
4221   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces

```

```

4222 }
4223 \protected\def\endmppattern{%
4224   \egroup
4225   \directlua{ luamplib.registerpattern(
4226     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
4227   )}%
4228   \endgroup
4229 }

```

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

```

4230 \def\mpfiginstancename{@mpfig}
4231 \protected\def\mpfig{%
4232   \begingroup
4233   \futurelet\nexttok\mplibmpfigbranch
4234 }
4235 \def\mplibmpfigbranch{%
4236   \ifx *\nexttok
4237     \expandafter\mplibprempfig
4238   \else
4239     \ifx [\nexttok
4240       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
4241     \else
4242       \expandafter\expandafter\expandafter\mplibmainmpfig
4243     \fi
4244   \fi
4245 }
4246 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
4247 \def\mplibmainmpfig{%
4248   \begingroup
4249   \mplibsetupcatcodes
4250   \mplibdomainmpfig
4251 }
4252 \long\def\mplibdomainmpfig#1\endmpfig{%
4253   \endgroup
4254   \directlua{
4255     local legacy = luamplib.legacyverbatimimtex
4256     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
4257     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
4258     luamplib.legacyverbatimimtex = false
4259     luamplib.everymplib["\mpfiginstancename"] = ""
4260     luamplib.everyendmplib["\mpfiginstancename"] = ""
4261     luamplib.process_mplibcode(
4262       "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===".." "..everyendmpfig.." endfig;",
4263       "\mpfiginstancename")
4264     luamplib.legacyverbatimimtex = legacy
4265     luamplib.everymplib["\mpfiginstancename"] = everympfig
4266     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
4267   }%
4268   \endgroup

```

```

4269 }
4270 \def\mplibprempfig#1{%
4271   \begingroup
4272   \mplibsetupcatcodes
4273   \mplibdoprempfig
4274 }
4275 \long\def\mplibdoprempfig#1\endmpfig{%
4276   \endgroup
4277   \directlua{
4278     local legacy = luamplib.legacyverbatim
4279     local everypfig = luamplib.everypfig["\mpfiginstancename"]
4280     local everyendmpfig = luamplib.everyendmpfig["\mpfiginstancename"]
4281     luamplib.legacyverbatim = false
4282     luamplib.everypfig["\mpfiginstancename"] = ""
4283     luamplib.everyendmpfig["\mpfiginstancename"] = ""
4284     luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
4285     luamplib.legacyverbatim = legacy
4286     luamplib.everypfig["\mpfiginstancename"] = everypfig
4287     luamplib.everyendmpfig["\mpfiginstancename"] = everyendmpfig
4288   }%
4289   \endgroup
4290 }
4291 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

4292 \unless\ifcsname ver@luamplib.sty\endcsname
4293   \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
4294   \protected\def\mplibcode{%
4295     \begingroup
4296     \futurelet\nexttok\mplibcodebranch
4297   }
4298   \def\mplibcodebranch{%
4299     \ifx [\nexttok
4300       \expandafter\mplibcodegetinstancename
4301     \else
4302       \global\let\currentmpinstancename\empty
4303       \expandafter\mplibcodeindeed
4304     \fi
4305   }
4306   \def\mplibcodeindeed{%
4307     \begingroup
4308     \mplibsetupcatcodes
4309     \mplibdocode
4310   }
4311   \long\def\mplibdocode#1\endmplibcode{%
4312     \endgroup
4313     \directlua{luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\currentmpinstancename")}%
4314     \endgroup
4315   }

```

```

4316 \protected\def\endmplibcode{\endmplibcode}
4317 \else

```

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

```

4318 \newenvironment{mplibcode}[1][{}]{%
4319   \xdef\currentmpinstancename{#1}%
4320   \mplibtmptoks{}\ltxdomplibcode
4321 }{}
4322 \def\ltxdomplibcode{%
4323   \begingroup
4324   \mplibsetupcatcodes
4325   \ltxdomplibcodeindeed
4326 }
4327 \def\mplib@mplibcode{mplibcode}
4328 \long\def\ltxdomplibcodeindeed#1\end#2{%
4329   \endgroup
4330   \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
4331   \def\mplibtemp@a{#2}%
4332   \ifx\mplib@mplibcode\mplibtemp@a
4333     \directlua{luamplib.process_mplibcode([===\the\mplibtmptoks===],"\currentmpinstancename")}%
4334     \end{mplibcode}%
4335   \else
4336     \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
4337     \expandafter\ltxdomplibcode
4338   \fi
4339 }
4340 \fi

```

User settings.

```

4341 \def\mplibshowlog#1{\directlua{
4342   local s = string.lower("#1")
4343   if s == "enable" or s == "true" or s == "yes" then
4344     luamplib.showlog = true
4345   else
4346     luamplib.showlog = false
4347   end
4348 }}
4349 \def\mpliblegacybehavior#1{\directlua{
4350   local s = string.lower("#1")
4351   if s == "enable" or s == "true" or s == "yes" then
4352     luamplib.legacyverbatiminput = true
4353   else
4354     luamplib.legacyverbatiminput = false
4355   end
4356 }}
4357 \def\mplibverbatim#1{\directlua{
4358   local s = string.lower("#1")
4359   if s == "enable" or s == "true" or s == "yes" then
4360     luamplib.verbatiminput = true
4361   else

```

```

4362     luamplib.verbatiminput = false
4363     end
4364 }}
4365 \newtoks\mplibtmptoks

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

4366 \ifcsname ver@luamplib.sty\endcsname
4367   \protected\def\everymplib{%
4368     \begingroup
4369     \mplibsetupcatcodes
4370     \mplibdoeverymplib
4371   }
4372   \protected\def\everyendmplib{%
4373     \begingroup
4374     \mplibsetupcatcodes
4375     \mplibdoeveryendmplib
4376   }
4377   \newcommand\mplibdoeverymplib[2][]{%
4378     \endgroup
4379     \directlua{
4380       luamplib.everymplib["#1"] = [===[\unexpanded{#2}]==]
4381     }%
4382   }
4383   \newcommand\mplibdoeveryendmplib[2][]{%
4384     \endgroup
4385     \directlua{
4386       luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]==]
4387     }%
4388   }
4389 \else
4390   \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
4391   \protected\def\everymplib#1{%
4392     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4393     \begingroup
4394     \mplibsetupcatcodes
4395     \mplibdoeverymplib
4396   }
4397   \long\def\mplibdoeverymplib#1{%
4398     \endgroup
4399     \directlua{
4400       luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]==]
4401     }%
4402   }
4403   \protected\def\everyendmplib#1{%
4404     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4405     \begingroup
4406     \mplibsetupcatcodes
4407     \mplibdoeveryendmplib
4408   }

```

```

4409 \long\def\mplibdoeveryendmplib#1{%
4410   \endgroup
4411   \directlua{
4412     luampLib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
4413   }%
4414 }
4415 \fi

```

TeX macros for dimen/color

```

4416 \def\mpdim#1{ runscript("luampLibdimen{#1}") }
4417 \def\mpcolor#1#\domplibcolor{#1}}
4418 \def\domplibcolor#1#2{ runscript("luampLibcolor{#1{#2}}") }

```

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

```

4419 \def\mplibnumbersystem#1{\directlua{
4420   local t = "#1"
4421   if t == "binary" then t = "decimal" end
4422   luampLib.numbersystem = t
4423 }}

```

Settings for .mp cache files.

```

4424 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4425 \def\mplibdomakenocache#1,{%
4426   \ifx\empty#1\empty
4427     \expandafter\mplibdomakenocache
4428   \else
4429     \ifx\stop#1\else
4430       \directlua{luampLib.noneedtoreplace["#1.mp"]=true}%
4431       \expandafter\expandafter\expandafter\mplibdomakenocache
4432     \fi
4433   \fi
4434 }
4435 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4436 \def\mplibdocancelnocache#1,{%
4437   \ifx\empty#1\empty
4438     \expandafter\mplibdocancelnocache
4439   \else
4440     \ifx\stop#1\else
4441       \directlua{luampLib.noneedtoreplace["#1.mp"]=false}%
4442       \expandafter\expandafter\expandafter\mplibdocancelnocache
4443     \fi
4444   \fi
4445 }
4446 \def\mplibcachedir#1{\directlua{luampLib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4447 \def\mplibtexttextlabel#1{\directlua{
4448   local s = string.lower("#1")
4449   if s == "enable" or s == "true" or s == "yes" then
4450     luampLib.texttextlabel = true

```

```

4451     else
4452         luamplib.texttextlabel = false
4453     end
4454 }}
4455 \def\mplibcodeinherit#1{\directlua{
4456     local s = string.lower("#1")
4457     if s == "enable" or s == "true" or s == "yes" then
4458         luamplib.codeinherit = true
4459     else
4460         luamplib.codeinherit = false
4461     end
4462 }}
4463 \def\mplibglobaltexttext#1{\directlua{
4464     local s = string.lower("#1")
4465     if s == "enable" or s == "true" or s == "yes" then
4466         luamplib.globaltexttext = true
4467     else
4468         luamplib.globaltexttext = false
4469     end
4470 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

4471 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4472 \def\mplibstarttoPDF#1#2#3#4{%
4473     \prependtomplibbox
4474     \hbox dir TLT \luamplibexternalizemaybe \bgroup
4475     \xdef\MPllx{#1}\xdef\MPlly{#2}%
4476     \xdef\MPurx{#3}\xdef\MPury{#4}%
4477     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4478     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4479     \parskip0pt%
4480     \leftskip0pt%
4481     \parindent0pt%
4482     \everypar{}%
4483     \setbox\mplibscratchbox\vbox\bgroup
4484     \noindent
4485 }
4486 \def\mplibstoptoPDF{%
4487     \par
4488     \egroup %
4489     \setbox\mplibscratchbox\hbox %
4490         {\hskip-\MPllx bp%
4491          \raise-\MPlly bp%
4492          \box\mplibscratchbox}%
4493     \setbox\mplibscratchbox\vbox to \MPheight
4494         {\vfill
4495          \hsize\MPwidth

```

```

4496 \wd\mplibscratchbox0pt%
4497 \ht\mplibscratchbox0pt%
4498 \dp\mplibscratchbox0pt%
4499 \box\mplibscratchbox}%
4500 \wd\mplibscratchbox\MPwidth
4501 \ht\mplibscratchbox\MPheight
4502 \box\mplibscratchbox
4503 \egroup
4504 }

```

Text items have a special handler.

```

4505 \def\mplibtexttext#1#2#3#4#5{%
4506 \begingroup
4507 \setbox\mplibscratchbox\hbox
4508 {\font\temp=#1 at #2bp%
4509 \temp
4510 #3}%
4511 \setbox\mplibscratchbox\hbox
4512 {\hskip#4 bp%
4513 \raise#5 bp%
4514 \box\mplibscratchbox}%
4515 \wd\mplibscratchbox0pt%
4516 \ht\mplibscratchbox0pt%
4517 \dp\mplibscratchbox0pt%
4518 \box\mplibscratchbox
4519 \endgroup
4520 }

```

Input luamplib.cfg when it exists.

```

4521 \openin0=luamplib.cfg
4522 \ifeof0 \else
4523 \closein0
4524 \input luamplib.cfg
4525 \fi

```

Code for $\LaTeX$, especially tagpdf

```

4526 \def\luamplibtagtextboxset#1#2{#2}
4527 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4528 \let\luamplibtagasgroupset\relax
4529 \let\luamplibtagasgroupput\luamplibtagtextboxset
4530 \let\luamplibexternalizemaybe\relax
4531 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4532

```

externalize functionality supports $\LaTeX$ PDF mode only.

```

4533 \def\mplibexternalize{%
4534 \newcount\luamplibexternalizecount
4535 \newattribute\luamplibexternalizeattr
4536 \chardef\luamplibexternalizealloc\allocationnumber
4537 \def\luamplibexternalizethisfigure{%
4538 \def\luamplibexternalizemaybe{attr \luamplibexternalizealloc =\luamplibexternalizecount}%

```

```

4539 }%
4540 \directlua{luamplib.externalize.setup()}%
4541 \def\luamplibexternalizedothis{true}%
4542 \aftergroup\mplibexternalizesuspend
4543 }
4544 \def\mplibexternalizenamename#1{\directlua{luamplib.externalize.MY_NAME=[==[#1]==]}}
4545 \def\mplibexternalizesuspend{\ifdefined\luamplibexternalizecount
4546 \directlua{luamplib.externalize.running = false}\fi}
4547 \def\mplibexternalizeresume{\ifdefined\luamplibexternalizecount
4548 \directlua{luamplib.externalize.running = true}\fi}
4549
4550 \ifcsname ver@tagpdf.sty\endcsname \else
4551 \ExplSyntaxOn
4552 \keys_define:nn{luamplib/tagging}
4553 {
4554 ,alt .code:n = { }
4555 ,actualtext .code:n = { }
4556 ,artifact .code:n = { }
4557 ,text .code:n = { }
4558 ,off .code:n = { }
4559 ,tag .code:n = { }
4560 ,adjust-BBox .code:n = { }
4561 ,tagging-setup .code:n = { }
4562 ,externalize .code:n = { \cs_set_nopar:Npn \luamplibexternalizedothis {#1} }
4563 ,externalize .default:n = { true }
4564 ,externalizedependson .code:n = { \cs_set_nopar:Npn \luamplibexternalizedependson {#1} }
4565 ,externalizemargin .code:n = { \cs_set_nopar:Npn \luamplibexternalizemargin {#1} }
4566 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4567 ,instancename .meta:n = { instance = {#1} }
4568 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4569 }
4570 \RenewDocumentCommand\mplibcode{0{}}
4571 {
4572 \tl_gclear:N \currentmpinstancename
4573 \keys_set:ne{luamplib/tagging}{#1}
4574 \mplibtmptoks{}\ltxdomplibcode
4575 }
4576 \RenewDocumentCommand\mpfig{s 0{}}
4577 {
4578 \group_begin:
4579 \keys_set_known:ne {luamplib/tagging} {#2}
4580 \IfBooleanTF{#1} { \mplibprempfig * }
4581 { \mplibmainmpfig }
4582 }
4583 \RenewDocumentCommand\usemplibgroup{0{ } m}
4584 {
4585 \group_begin:
4586 \keys_set_known:ne {luamplib/tagging} {#1}
4587 \luamplib@externalized{luamplib.group.#2}

```

```

4588 \int_compare:nNt {\l_tmpa_int} = {\c_zero_int}
4589 {
4590 \prependtomplibbox\hbox dir~TLT~\luamplibexternalizemaybe \bgroup
4591 \csname luamplib.group.#2\endcsname
4592 \egroup
4593 }
4594 \group_end:
4595 }
4596 \cs_set_eq:NN \mplibalttext \use_none:n
4597 \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.

```

4598 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4599 \ExplSyntaxOff
4600 \endinput\fi
4601 \ExplSyntaxOn
4602 \tl_new:N \l__luamplib_tag_envname_tl
4603 \tl_new:N \l__luamplib_tag_alt_tl
4604 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4605 \tl_new:N \l__luamplib_tag_actual_tl
4606 \tl_new:N \l__luamplib_tag_struct_tl
4607 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4608 \bool_new:N \l__luamplib_tag_usetext_bool
4609 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4610 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4611 \tl_new:N \l__luamplib_tag_bbox_draw_tl
4612 \tl_new:N \l__luamplib_BBox_llx_tl
4613 \tl_new:N \l__luamplib_BBox_lly_tl
4614 \tl_new:N \l__luamplib_BBox_urx_tl
4615 \tl_new:N \l__luamplib_BBox_ury_tl
4616 \msg_new:nnn {luamplib}{figure-text-reuse}
4617 {
4618 tex-text~box~#1~probably~is~incorrectly~tagged.~
4619 Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4620 Check~the~resulting~PDF.
4621 }
4622 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4623 {
4624 mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4625 Using~mplibgroup~with~text~mode~is~not~recommended.~
4626 Check~the~resulting~PDF.
4627 }
4628 \msg_new:nnn {luamplib}{alt-text-missing}
4629 {
4630 Alternate~text~for~#1~is~missing.~

```

```

4631 Using~the~default~value~'#2'~instead.
4632 }

```

Sockets for tex-text boxes.

```

4633 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4634 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4635 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4636 {

```

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.

```

4637 \bool_if:NTF \l__luamplib_tag_usetext_bool
4638 {
4639   \tag_mc_end_push:
4640   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4641   \cs_gset_nopar:cpe {luamplib.taggedbox.#1} {\tag_get:n{struct_num}}

```

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

```

4642   \tag_mc_begin:n{tag=text}
4643   #2
4644   \tag_mc_end:
4645   \tag_struct_end:
4646   \tag_mc_begin_pop:n{ }
4647 }
4648 {
4649   \tag_suspend:n{\luamplibtagtextboxset}
4650   #2
4651   \tag_resume:n{\luamplibtagtextboxset}
4652 }
4653 }
4654 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4655 {
4656   \bool_lazy_and:nnTF
4657   { \l__luamplib_tag_usetext_bool }
4658   { \cs_if_free_p:c {luamplib.nottaggedbox.#1} }
4659   {
4660     \tag_resume:n{\mplibputtextbox}
4661     \tag_mc_end:
4662     \cs_if_exist:cTF {luamplib.taggedbox.#1}
4663     {
4664       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4665       #2
4666       \cs_undefine:c {luamplib.taggedbox.#1}
4667     }
4668     {
4669       \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
4670       \tag_mc_begin:n{ }
4671       \int_set:Nn \l_tmpa_int {#1}
4672       \tag_mc_reset_box:N \l_tmpa_int
4673       #2

```

```

4674     \tag_mc_end:
4675   }
4676   \tag_mc_begin:n{artifact}
4677 }
4678 {
4679   \int_set:Nn \l_tmpa_int {#1}
4680   \tag_mc_reset_box:N \l_tmpa_int
4681   #2
4682 }
4683 }
4684 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4685 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4686 \cs_set_nopar:Npn \luamplibtagtextboxset
4687 {
4688   \tag_socket_use:nnn{luamplib/texttext/set}
4689 }

```

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

```

4690 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4691 {
4692   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
4693   \bool_set_false:N \l__luamplib_tag_usetext_bool
4694   \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
4695   \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4696   \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4697 }
4698 \sys_if_output_pdf:TF
4699 {
4700   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4701   {
4702     \pdfextension save\relax
4703     \vbox to 0pt{\vss
4704       \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4705         \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4706       \kern #3bp}
4707     \pdfextension restore\relax
4708   }
4709 }
4710 {
4711   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4712   {
4713     \special{pdf:btrans~matrix~#4~#2~#3}
4714     \vbox to 0pt{\vss\hbox bdir0 to 0pt{
4715       \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4716     \special{pdf:etrans}
4717   }
4718 }

```

TODO: Not sure whether asgroup/mplibgroup with text mode will be tagged correctly. Prob-

ably not. At least, this will raise a warning.

```
4719 \cs_set_nopar:Npn \luamplibtagasgroupset
4720 {
4721   \bool_set_false:N \l__luamplib_tag_usetext_bool
4722 }
4723 \cs_set_nopar:Npn \luamplibtagasgroupput
4724 {
4725   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4726   \tag_socket_use:nnn{\luamplib/mplibgroup/put}
4727 }
```

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

```
4728 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4729 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4730 {
4731   \cs_if_free:cT {luamplib.mplibgroup.text.#1}
4732   {
4733     \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4734     \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4735   }
4736   \tag_mc_end:
4737   \tag_mc_begin:n{tag=text}
4738   #2
4739   \tag_mc_end:
4740   \tag_mc_begin:n{artifact}
4741 }
4742 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}
```

A macro for BBox attribute

```
4743 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4744 {
4745   \tl_set:Ne \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4746   \tex_savepos:D
4747   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4748   \tl_set:Ne \l__luamplib_BBox_llx_tl
4749   { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4750   \tl_set:Ne \l__luamplib_BBox_lly_tl
4751   { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4752   \tl_set:Ne \l__luamplib_BBox_urx_tl
4753   { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4754   \tl_set:Ne \l__luamplib_BBox_ury_tl
4755   { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4756   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4757   {
4758     \int_zero:N \l_tmpa_int
4759     \tl_map_inline:nn
4760     {
4761       \l__luamplib_BBox_llx_tl
4762       \l__luamplib_BBox_lly_tl
4763       \l__luamplib_BBox_urx_tl
```

```

4764     \l__luamplib_BBox_ury_tl
4765 }
4766 {
4767     \int_incr:N \l_tmpa_int
4768     \tl_set:Ne ##1
4769     {
4770         \fp_eval:n
4771         {
4772             ##1
4773             +
4774             \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4775         }
4776     }
4777 }
4778 }
4779 \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4780 {
4781     /O /Layout /BBox [
4782         \l__luamplib_BBox_llx_tl\c_space_tl
4783         \l__luamplib_BBox_lly_tl\c_space_tl
4784         \l__luamplib_BBox_urx_tl\c_space_tl
4785         \l__luamplib_BBox_ury_tl
4786     ]
4787 }
4788 \bool_if:NT \l__tag_graphic_debug_bool
4789 {
4790     \iow_log:e
4791     {
4792         luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4793         \l__luamplib_BBox_llx_tl\c_space_tl
4794         \l__luamplib_BBox_lly_tl\c_space_tl
4795         \l__luamplib_BBox_urx_tl\c_space_tl
4796         \l__luamplib_BBox_ury_tl
4797     }
4798     \sys_if_output_pdf:TF
4799     {
4800         \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
4801         {
4802             \pdfextension save\relax
4803             \opacity_select:n{0.5} \color_select:n{red}
4804             \pdfextension literal~text
4805             {
4806                 \l__luamplib_BBox_llx_tl\c_space_tl
4807                 \l__luamplib_BBox_lly_tl\c_space_tl
4808                 \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4809                 \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4810                 re~f
4811             }
4812             \pdfextension restore\relax

```

```

4813     }
4814   }
4815   {
4816     \tl_set:Nc \l__luamplib_tag_bbox_draw_tl
4817     {
4818       \special{pdf:bcontent}
4819       \opacity_select:n{0.5} \color_select:n{red}
4820       \special{pdf:code~
4821         1~0~0~1~
4822         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4823         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4824         cm
4825       }
4826       \special{pdf:code~
4827         \l__luamplib_BBox_llx_tl\c_space_tl
4828         \l__luamplib_BBox_lly_tl\c_space_tl
4829         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4830         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4831         re~f
4832       }
4833       \special{pdf:econtent}
4834     }
4835   }
4836 }
4837 }

```

Sockets for main process

```

4838 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4839 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4840 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4841 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4842 {
4843   \tag_mc_end_push:
4844   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4845   {
4846     \tl_if_empty:eTF{#1}
4847     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4848     { \tl_set:Nc \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4849     \msg_warning:nnVV{luamplib}{alt-text-missing}
4850     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4851   }
4852   \tag_struct_begin:n
4853   {
4854     tag=\l__luamplib_tag_struct_tl,
4855     alt=\l__luamplib_tag_alt_tl,
4856   }
4857   \tag_mc_begin:n{}
4858 }
4859 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}

```

```

4860 {
4861   \cs_if_eq:NNT \luamplibexternalizemaybe \relax
4862   {
4863     \__luamplib_tag_bbox_attribute:n {#1}
4864   }
4865   #2
4866   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4867   \tag_mc_end:
4868   \tag_struct_end:
4869   \tag_mc_begin_pop:n{ }
4870 }
4871 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4872 {
4873   \tag_mc_end_push:
4874   \tag_struct_begin:n
4875   {
4876     tag=Span,
4877     actualtext=\l__luamplib_tag_actual_tl,
4878   }
4879   \tag_mc_begin:n{ }
4880 }
4881 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4882 {
4883   #2
4884   \tag_mc_end:
4885   \tag_struct_end:
4886   \tag_mc_begin_pop:n{ }
4887 }
4888 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4889 {
4890   \tag_mc_end_push:
4891   \tag_mc_begin:n{artifact}
4892 }
4893 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4894 {
4895   #2
4896   \tag_mc_end:
4897   \tag_mc_begin_pop:n{ }
4898 }

```

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

```

4899 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4900 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4901 {
4902   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4903   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4904 }
4905 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}

```

```

4906 {
4907   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4908   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4909   \prependtomplibbox \mplibnoforcehmode
4910   \mode_if_vertical:T { \noindent \aftergroup\par }
4911 }
4912 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4913 {
4914   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4915   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4916 }
4917 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4918 {
4919   \bool_set_true:N \l__luamplib_tag_usetext_bool
4920   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4921   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4922   \prependtomplibbox \mplibnoforcehmode
4923   \mode_if_vertical:T { \noindent \aftergroup\par }
4924 }
4925 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4926 {
4927   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4928   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4929 }
4930 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```

4931 \keys_define:nn{luamplib/tagging}
4932 {
4933   ,alt .code:n =
4934   {
4935     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4936     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4937   }
4938   ,actualtext .code:n =
4939   {
4940     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4941     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4942   }
4943   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4944   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4945   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4946   ,tag .code:n =
4947   {
4948     \str_case:nnF {#1}
4949     {
4950       {false} { \keys_set:nn {luamplib/tagging} {off} }
4951       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }

```

```

4952   }
4953   {
4954     \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4955     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4956   }
4957 }
4958 ,adjust-BBox .code:n =
4959 {
4960   \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4961   \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4962 }
4963 ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4964 }
4965 \keys_define:nn {luamplib/instance}
4966 {
4967   ,externalize .code:n = { \cs_set_nopar:Npn \luamplibexternalizedothis {#1} }
4968   ,externalize .default:n = { true }
4969   ,externalizedependson .code:n = { \cs_set_nopar:Npn \luamplibexternalizedependson {#1} }
4970   ,externalizemargin .code:n = { \cs_set_nopar:Npn \luamplibexternalizemargin {#1} }
4971   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4972   ,instancename .meta:n = { instance = {#1} }
4973   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4974 }

```

Redefine our macros

```

4975 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4976 {
4977   \prependtomplibbox
4978   \hbox dir~TLT~\luamplibexternalizemaybe \bgroup
4979     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4980     \xdef\MPllx{#1}\xdef\MPlly{#2}%
4981     \xdef\MPurx{#3}\xdef\MPury{#4}%
4982     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4983     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4984     \parskip0pt
4985     \leftskip0pt
4986     \parindent0pt
4987     \everypar{}%
4988     \setbox\mplibscratchbox\vbox\bgroup
4989       \tag_suspend:n{\mplibstarttoPDF}
4990       \noindent
4991 }
4992 \cs_set_nopar:Npn \mplibstoptoPDF
4993 {
4994   \par
4995   \egroup
4996   \setbox\mplibscratchbox\hbox
4997     {\hskip-\MPllx bp
4998     \raise-\MPlly bp

```

```

4999      \box\mplibscratchbox}%
5000      \setbox\mplibscratchbox\vbox to \MPheight
5001      {\vfill
5002       \hsize\MPwidth
5003       \wd\mplibscratchbox\pt
5004       \ht\mplibscratchbox\pt
5005       \dp\mplibscratchbox\pt
5006       \box\mplibscratchbox}%
5007      \wd\mplibscratchbox\MPwidth
5008      \ht\mplibscratchbox\MPheight
5009      \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
5010      \egroup
5011    }
5012    \RenewDocumentCommand\mplibcode{0{}}
5013    {
5014      \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
5015      \tl_gclear:N \currentmpinstancename
5016      \keys_set:known:neN {luamplib/tagging} {#1} \l_tmpa_tl
5017      \keys_set:nV {luamplib/instance} \l_tmpa_tl
5018      \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
5019      \tag_socket_use:n{luamplib/figure/init}
5020      \mplibtmptoks{}\ltxdomplibcode
5021    }
5022    \RenewDocumentCommand\mpfig{s 0{}}
5023    {
5024      \group_begin:
5025      \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
5026      \keys_set:known:neN {luamplib/tagging} {#2} \l_tmpa_tl
5027      \keys_set:nV {luamplib/instance} \l_tmpa_tl
5028      \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
5029      \tag_socket_use:n{luamplib/figure/init}
5030      \IfBooleanTF{#1} { \mplibprempfig * }
5031                      { \mplibmainmpfig }
5032    }
5033    \RenewDocumentCommand\usemplibgroup{0{ } m}
5034    {
5035      \group_begin:
5036      \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
5037      \keys_set:known:neN {luamplib/tagging} {#1} \l_tmpa_tl
5038      \keys_set:nV {luamplib/instance} \l_tmpa_tl
5039      \tag_socket_use:n{luamplib/figure/init}
5040      \luamplib@externalized{luamplib.group.#2}
5041      \int_compare:nNnT {\l_tmpa_int} = {\c_zero_int}
5042      {
5043        \prependtomplibbox\hbox dir~TLT~\luamplibexternalizemaybe \bgroup
5044        \tag_socket_use:nn{luamplib/figure/begin}{#2}
5045        \setbox\mplibscratchbox\hbox\bgroup
5046        \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
5047        \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}

```

```

5048     \egroup
5049     \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
5050     \egroup
5051   }
5052   \group_end:
5053 }

```

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

```

5054 \cs_new_nopar:Npn \mplibalttext #1
5055 {
5056   \tl_set:Nc \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
5057 }
5058 \cs_new_nopar:Npn \mplibactualtext #1
5059 {
5060   \tl_set:Nc \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
5061 }
5062 \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.