

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/09/22 V2.44.0

Abstract

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

Contents

1	Documentation	2
1.1	$\TeX$	3
1.1.1	Forcing horizontal mode	3
1.1.2	Insert some code into every code chunk	3
1.1.3	Choosing a METAPOST format	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	<code>\mplibglobaltexttext</code>	6
1.1.10	Separate METAPOST instances	6
1.1.11	Verbatim input	7
1.1.12	$\TeX$ dimen	7
1.1.13	$\TeX$ color	7
1.1.14	<code>\mpfig</code> , <code>\endmpfig</code>	8
1.1.15	About cache files	9
1.1.16	About figure box metric	9
1.1.17	<code>luamplib.cfg</code>	9
1.1.18	Tagged PDF	9
1.1.19	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	13

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	34
1.3.3	Running a METAPOST code	35
1.3.4	Registering a tiling pattern	36
1.3.5	Registering a form XObject	36
2	Implementation	36
2.1	Lua module	36
2.2	T _E Xpackage	125
3	The GNU GPL License v2	147

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, cannot properly render the isolated or knockout effect.

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

1.1.4 Choosing a number system

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

1.1.5 Showing METAPOST log

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

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

1.1.6 verbatimtex Legacy behavior

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

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

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

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

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



diameter: 22.62764bp.

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

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

ABC **DEF GHI**

1.1.7 $\TeX$ -text labels

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

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

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

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

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

1.1.8 Inherit METAPOST code

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

1.1.9 \mplibglobaltexttext{enable|disable}

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

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



1.1.10 Separate METAPOST instances

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

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

The behavior is as follows.

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

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

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

1.1.11 Verbatim input

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

1.1.12 $\TeX$ dimen

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

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



1.1.13 $\TeX$ color

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

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

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

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



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

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

1.1.14 `\mpfig ... \endmpfig`

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

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

and the starred version is roughly the same as follows:

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

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

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



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

1.1.15 About cache files

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

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

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

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

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

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

1.1.16 About figure box metric

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

1.1.17 luamplib.cfg

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

1.1.18 Tagged PDF

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

⁸Beware that this bypasses 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:

`\mplibexternalizenamename{ $\langle$ prefix $\rangle$ }` The filename of external images starts with $\langle$ jobname $\rangle$ -mplib-figure by default, which you can change by this command. For instance, `\mplibexternalizenamename{figures/metapost-figure}` will generate `metapost-figure-*.pdf` files in the `figures` sub-directory (the directory 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 $\LaTeX$, the world of externalization will start anew from the very beginning.

$\backslash\text{mplibexternalizesuspend}$ The operation of ‘externalize’ functionality will be halted. This command does not respect $\TeX$ ’s group scope.

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

Options to mplibcode environment, and to the $\backslash\text{mpfig}$ and $\backslash\text{usemplibgroup}$ commands:

$\text{externalize}=\langle boolean \rangle$ By the option **externalize** (shortcut of $\text{externalize}=\text{true}$), external image(s) for the specific METAPOST code chunk will be generated even if ‘externalize’ feature is suspended.

On the other hand, when **externalize=false** is given, external image(s) for the specific code chunk will not be generated even if ‘externalize’ feature is enabled.

$\text{externalizedependson}=\{\langle number \rangle, \dots\}$ If you modify a METAPOST code chunk which is dependent on previous code chunks (this occurs when $\backslash\text{mplibcodeinherit}$ 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 chunks by using this option, for instance $\text{externalizedependson}=\{2, 3\}$.

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

$\text{externalizemargin}=\langle dimen \rangle$ To address the cutting-off problem (see below the limitations), 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 $\TeX$ boxes remains the same.

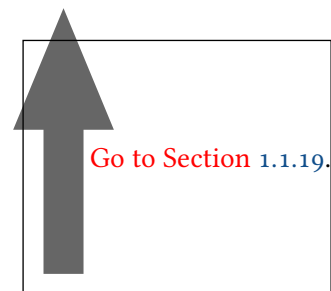
Known limitations

- The ‘externalize’ feature currently supports only $\LaTeX$ 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 before, externalizemargin option is provided to address this problem.
- Hyperlinks in $\text{btex} \dots \text{etex}$ (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 (especially 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.

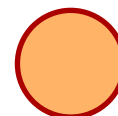
1.2.3 Fill and stroke colors

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`

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

(*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
  withhplibcolors ("orange!60", 2/3red) ;
```

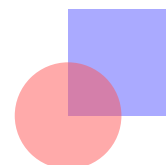


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

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.

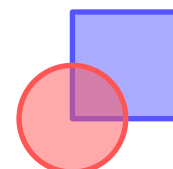
```
\mpfig
fill unitsquare scaled 40
  withcolor 1/3[blue,white]
  withtransparency (1, 0.5)      % or ("normal", 0.5)
;
fill fullcircle scaled 40
  withcolor 1/3[red,white]
  withtransparency (1, 0.5)
;
\endmpfig
```



1.2.5 Fill and stroke transparency

By analogy with the macro `withhplibcolors` (see above § 1.2.3), the macro `withhplibopacities` is also provided. The syntax is `withhplibopacities` (*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.

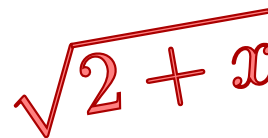
```
\mpfig
pickup pencircle scaled 2;
filldraw unitsquare scaled 40
  withcolor 1/3[blue,white]
  withhplibopacities (1, 1/2, 1)  % or ("normal", 1/2, 1)
;
filldraw fullcircle scaled 40
  withcolor 1/3[red,white]
  withhplibopacities (1, 1/2, 1)
;
\endmpfig
```



1.2.6 Text outline as a text image

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

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



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

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

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
```

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.

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.

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*).

A simple example:

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



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

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

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
```

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.

```

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))

withshadingfactor `<numeric>` Multiplier of the radii, default value being 1.2. This is no-op in linear mode.

withshadingtransform `<string>` where `<string>` 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 (*<pair>*, *<pair>*) 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 (*<pair>*, *<pair>*) 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.

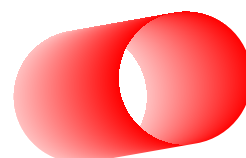
withshadingfraction *<numeric>* Fractional number of each shading step, and so only meaningful within `withshadingstep`.

withshadingcolors (*<color expr>*, *<color expr>*) Starting and ending colors, default value being (white, black). String-type argument is regarded as the color expression of $\TeX$ side.

withshadingdomain *<pair>* 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 (*<boolean>*, *<boolean>*) 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
```



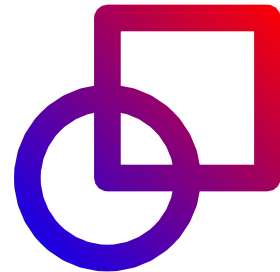
withshadingmatrix *<string>* METAPOST code for transformation of shading, such as "`xscaled 1.2 yscaled 0.8`"; or six numerics separated by spaces, such as "`1.2 0 0 0.8 0 0`" (xx, yx, xy, yy, x, and y-part respectively).

withshadingstroke *<string>* When the shading object is a *<path>*, 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 *<picture>*, 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 `<path> | <picture> withfademethod <string>`, 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 (`<numeric>`, `<numeric>`) sets the starting opacity and the ending opacity, default value being (1,0). '1' denotes full color; '0' full transparency.

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

withfadecenter is a synonym of `withfadevector`.

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

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

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

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

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

withfadebbox ($\langle pair \rangle$, $\langle pair \rangle$) sets the bounding box (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 $\langle numeric \rangle$ 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

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

An example:

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

¹⁵`withpattern` is an operator virtually the same as `withmppattern`, but the former forces a METAPOST picture. Therefore you cannot but use `draw` command with `withpattern` operator. On the other hand, $\langle cyclic path \rangle$ `withmppattern  $\langle string \rangle$` works as intended only with `fill` or `filldraw` command.

Table 1: options for \mppattern

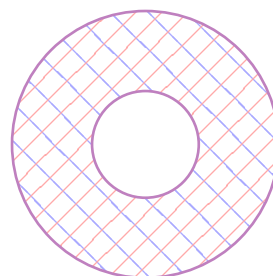
Key	Value Type	Explanation
xstep	<i>number</i>	horizontal spacing between pattern cells
ystep	<i>number</i>	vertical spacing between pattern cells
xshift	<i>number</i>	horizontal shifting of pattern cells
yshift	<i>number</i>	vertical shifting of pattern cells
bbox	<i>table</i> or <i>string</i>	llx, 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
  \mplibdrawglyph image(
    draw fullcircle scaled 100;
    draw reverse fullcircle scaled 40;
  )
  \withmppattern "mypatt"
  \withpen pencircle scaled 1
  \withcolor \mpcolor{red!50!blue!50} ;
\endmpfig

```



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

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

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

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

```

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

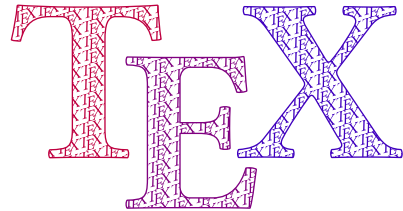
```

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

```

\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}

```



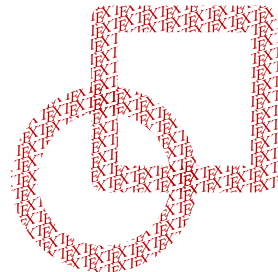
After v2.42.6, we provide an optional macro for both colored and uncolored patterns:

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: $\langle picture \rangle \mid \langle path \rangle \text{ asgroup } \langle string \rangle$, 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 $\TeX$ code or in other METAPOST code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide $\TeX$ and METAPOST macros as follows:

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

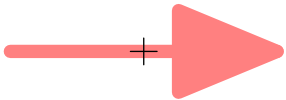
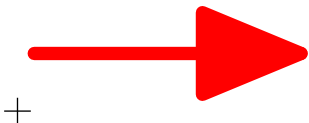
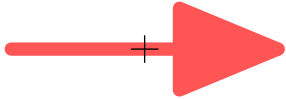
\usemplibgroup $\{ \langle name \rangle \}$ is a $\TeX$ command to reuse an XObject of the name once used. Note that the position of the XObject will be origin-based: in other words, lower left corner of the bounding box will be shifted to the origin.

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

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

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, including Mac OS Preview and Foxit Editor, do not interpret PDF coordinates properly. If that happens, consider using other PDF viewer such as Adobe Acrobat or MuPDF. An example:

```
\mpfig*
picture pic[];
pic1 = image(
```

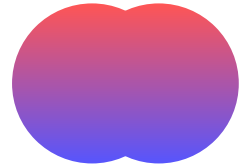
```

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

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

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

```



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

```

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

```



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

```

\mplibgroup{myshading}[asgroup="wrapped"]
    \mpfig
        draw pic2 ;
    \endmpfig
\endmplibgroup

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

```



Table 2: options for `\mplibgroup`

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

* in string type, numbers are separated by spaces

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

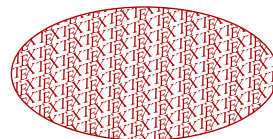
```

\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.

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.

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

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

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

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

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

1.2.17 Masking

Using the command `withmaskinggroup`, the `mplibgroup` (see above § 1.2.16) generated by the option `asgroup="masking"` (see Table 2) can be utilized as a masking transparency group upon a picture or a path object. The syntax is `<picture> | <path> withmaskinggroup <string>`, 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 <color expr>` sets the background color of the masking group. 0 denotes full transparency (masking invisibly); 1, full color.²⁰

An example:

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

¹⁸Note that some PDF renderers such as Mac OS Preview 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"` or `colorspace="/DeviceCMYK"` given to the masking transparency group. This however we do not recommend as the luminosity is difficult to understand intuitively.

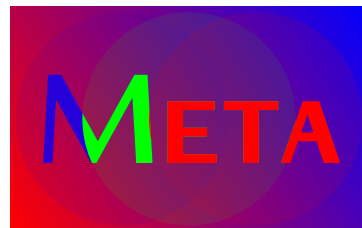
```

        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.

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

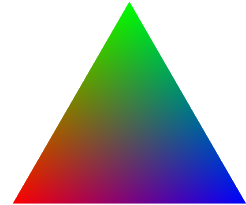
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates).

The third argument is the color for the new vertex.

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

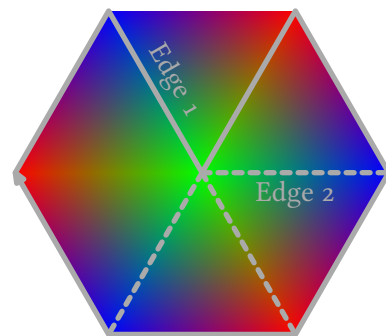
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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



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

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



1.2.19 Lattice-form Gouraud-shaded triangle mesh shading

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

Optional macros for lattice-form shading:

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

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

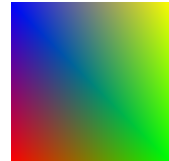
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

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

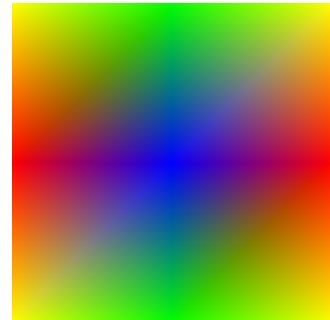
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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

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



```
\mpfig
  color yellow;
  yellow = (1,1,0);
  draw unitsquare scaled 2u
    withshadingmethod "lattice"
    withlatticevertexesperrow 3
    withlatticevertexesdata (
      (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 (*<number>*, *<path> | <string>*, *<color>*, *<color>*) The new patch attached to the previous one. This optional macro requires *withcoonspatchinit* specified beforehand.

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

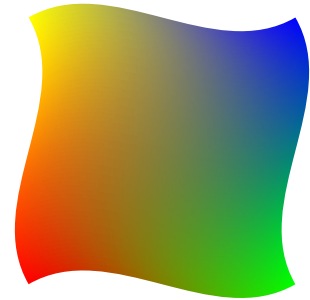
The second argument shall be a path that has four vertices, or a string of sixteen numerics separated by spaces (xpart postcontrol 1 of p, ypart postcontrol 1 of p, ..., xpart precontrol

0 of p, ypart precontrol 0 of p). The orientation of the new path should be reversed from the previous one, and it is assumed that the first segment of the new path coincides with the edge segment just mentioned.

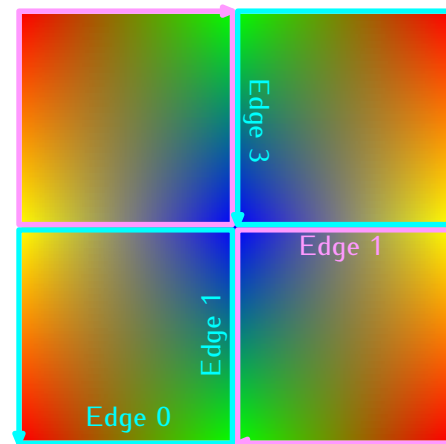
Remaining arguments are two color expressions for the new vertices.

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

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



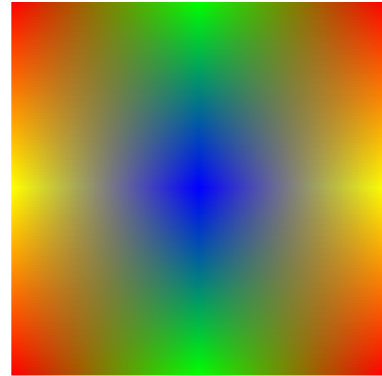
```
\mpfig
u := 80 ;
path p;
p = unitsquare scaled u;
def fsquare (expr n) =
  ( point n of p --
    point n+1 of p --
    point n+2 of p --
    point n+3 of p --
    cycle )
enddef;
def rsquare (expr n) =
  ( point n of p --
    point n-1 of p --
    point n-2 of p --
    point n-3 of p --
    cycle )
enddef;
fill p scaled 2
  withshadingmethod "coons"
  withcoonspatchinit
    (fsquare(0), red, green, blue, yellow)
  withcoonspatchnext
    (1, rsquare(0) shifted (u,0), yellow, red)
  withcoonspatchnext
    (1, fsquare(0) shifted (u,u), red, green)
  withcoonspatchnext
    (3, rsquare(2) shifted (0,u), yellow, red) ;
\endmpfig
```



point 1 of patch₁ = point 0 of patch₂

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

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

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

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

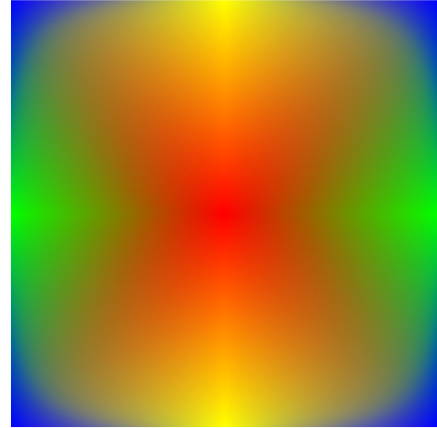
An example to show the effect of tensor-product patch mesh shading:

```
\mpfig
```

```

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

```



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

1.3 Lua

1.3.1 runscript ...

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

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

1.3.2 Accessing METAPOST variables

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

```

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

```

Table 3: elements in luamplib table (partial)

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

```

\end{mplibcode}

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

```

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

```

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

```

3.0

1.3.3 Running a METAPOST code

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

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

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

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

1.3.4 Registering a tiling pattern

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

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

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 $\TeX$ (not METAPOST) code, please make sure that both of the $\TeX$ 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 $\TeX$ macro ‘`luamplib.group.<name>`’. If you need the `boxresource` index, inspect this macro using `token.get_macro` function.

2 Implementation

2.1 Lua module

```
1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.44.0",
5   date      = "2026/09/22",
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. $\text{Con}\TeX\text{t}$ uses `metapost`.

```
9 luamplib      = luamplib or { }
10 local luamplib = luamplib
```

²¹In DVI mode, $\TeX$ 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, $\TeX$ 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`.

```

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 texrun toks 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 iioopen = io.open
70

```

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

```

71 local file = file or { }
72 local replacesuffix = file.replacesuffix or function(filename, suffix)
73   return (filename:gsub("%.[%a%d]+$", "")) .. "." .. suffix
74 end
75 local is_writable = file.is_writable or function(name)
76   if lfsisdir(name) then
77     name = name .. "/_luam_plib_temp_file_"
78     local fh = iioopen(name, "w")
79     if fh then
80       fh:close(); os.remove(name)
81       return true
82     end
83   end
84 end
85 local mk_full_path = lfs.mkdirp or lfs.mkdirs or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\n/]+)") do
88     full = full .. sub
89     lfsmkdir(full)
90   end
91 end
92

```

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

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

```

93 local cachedir
94 local function outputdir ()
95   if lfstouch then
96     for i,v in ipairs{'TEXMFVAR', 'TEXMF_OUTPUT_DIRECTORY', '.', 'TEXMFOUTPUT'} do
97       local var = i == 3 and v or kpse.var_value(v)
98       if var and var ~= "" then
99         for _,vv in ipairs(var:explode(os.type == "unix" and ":" or ";")) do
100           local dir = format("%s/%s", vv, "luamplib_cache")

```

```

101         if not lfsisdir(dir) then
102             mk_full_path(dir)
103         end
104         if is_writable(dir) then
105             cachedir = dir; return cachedir
106         end
107     end
108 end
109 end
110 end
111 cachedir = "."; return cachedir
112 end
113 function luamplib.getcachedir(dir)
114     dir = dir:gsub("##", "#")
115     dir = dir:gsub("^~",
116         os.type == "windows" and os.getenv("UserProfile") or os.getenv("HOME"))
117     if lfstouch and dir then
118         if lfsisdir(dir) then
119             if is_writable(dir) then
120                 cachedir = dir
121             else
122                 warn("Directory '%s' is not writable!", dir)
123             end
124         else
125             warn("Directory '%s' does not exist!", dir)
126         end
127     end
128 end

```

Some basic METAPOST files not necessary to make cache files.

```

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

```

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 b_{tex} occurring in them is just a word, and pairing it with the next real e_{tex} 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: b_{tex} and verbatim_{tex} 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 e_{tex} matters—comments are not special there, exactly as in METAPost. Finally, escapepercent tells that a percent sign preceded by a backslash is not the start of a comment.

```

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

```

```

187     local block = data:sub(s,e)
188     block = block == "btex" and action.btex
189           or block == "verbatimtex" and action.verbatimtex
190     local es, ee
191     if block then es, ee = data:find(etex_pat, e+1) end
192     if es then
193         res[n+1], res[n+2] = data:sub(pos,s-1), block(trim(data:sub(e+1,es-1)))
194         n, pos, i, count = n+2, ee+1, ee+1, count+1
195     else
196         i = e + 1 -- no etex in sight: not a block after all
197     end
198 end
199 end
200 if pos == 1 then return data, count end
201 res[n+1] = data:sub(pos)
202 return tableconcat(res), count
203 end
204
205 local replace_texblock = {
206     btex      = function(str) return format("btex %s etex ", str) end, -- space
207     verbatimtex = function(str) return format("verbatimtex %s etex;", str) end, -- semicolon
208 }
209

```

Function `luamplib.finder`

```

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

```

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

```

213 local function replaceformatmp(file,newfile,ofmodify)
214     local fh = ioopen(file,"r")
215     if not fh then return file end
216     local data = fh:read("*all"); fh:close()
217     fh = ioopen(newfile,"w")
218     if not fh then return file end
219     fh:write(
220         "let normalinfont = infont;\n",
221         "primarydef str infont name = rawtexttext(str) enddef;\n",
222         data,
223         "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
224         "vardef Fexp_(expr x) = rawtexttext(\"$^{\"&decimal x&\"}$\") enddef;\n",
225         "let infont = normalinfont;\n"
226     ); fh:close()
227     lfstouch(newfile,currenttime,ofmodify)
228     return newfile
229 end
230 local function replaceinputmpfile (name,file)
231     local ofmodify = lfsattributes(file,"modification")
232     if not ofmodify then return file end

```

```

233 local newfile = name:gsub("%W", "_")
234 newfile = format("%s/luamplib_input_%s", cachedir or outputdir(), newfile)
235 if newfile and luamplibtime then
236     local nf = lfsattributes(newfile)
237     if nf and nf.mode == "file" and
238         ofmodify == nf.modification and luamplibtime < nf.access then
239         return nf.size == 0 and file or newfile
240     end
241 end
242 if name == "format.mp" then return replaceformatmp(file, newfile, ofmodify) end
243 local fh = ioopen(file, "r")
244 if not fh then return file end
245 local data = fh:read("*all"); fh:close()

```

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

```

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

```

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

```

263 local mpkpse
264 do
265     local exe = 0
266     while arg[exe-1] do
267         exe = exe-1
268     end
269     mpkpse = kpse.new(arg[exe], "mpost")
270 end
271 local special_ftype = {
272     pfb = "type1 fonts",
273     enc = "enc files",
274 }
275 function luamplib.finder (name, mode, ftype)
276     if mode == "w" then

```

```

277     if name and name ~= "mpout.log" then
278         kpse.record_output_file(name) -- recorder
279     end
280     return name
281 else
282     ftype = special_ftype[ftype] or ftype
283     local file = mpkpse.find_file(name, ftype)
284     if file then
285         if lfstouch and ftype == "mp" and not noneedtoreplace[name] and not noneedtoreplace["*.mp"] then
286             file = replaceinputmpfile(name, file)
287         end
288     else
289         file = mpkpse.find_file(name, name:match("%a+$"))
290     end
291     if file then
292         kpse.record_input_file(file) -- recorder
293     end
294     return file
295 end
296 end
297 end
298

```

For the main function: process

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

```

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

```

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

```

303 luamplib.codeinherit = false
304 local mplibinstances = {}
305 luamplib.instances = mplibinstances
306 local has_instancename = false
307
308 local process
309 do
310     local function reporterror (result, prevlog)
311         if not result then
312             err("no result object returned")
313         else
314             local t, e, l = result.term, result.error, result.log

```

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

```

315         local log = l or t or "no-term"
316         log = log:gsub("%(Please type a command or say `end'%)", ""):gsub("\n+", "\n")
317         if result.status > 0 then
318             local first = log:match("(-\n! .-)\n! "
319             if first then
320                 termorlog("term", first)

```

```

321     termorlog("log", log, "Warning")
322   else
323     warn(log)
324   end
325   if result.status > 1 then
326     err(e or "see above messages")
327   end
328   elseif prevlog then
329     log = prevlog..log

```

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

```

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

```

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

```

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

```

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

```

346   make_text   = luamplib.maketext,
347   run_script  = luamplib.runscript,
348   math_mode   = luamplib.numbersystem,
349   job_name    = tex.jobname,
350   random_seed = math.random(4095),
351   utf8_mode   = true,
352   extensions  = 1,
353 }

```

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

```

354 local preamble = tableconcat{
355   format(luamplib.preambles.preamble, replacesuffix(name,"mp")),
356   luamplib.preambles.mplibcode,
357   luamplib.legacyverbatim and luamplib.preambles.legacyverbatim or "",
358   luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
359 }
360 local result, log

```

```

361     if not mpx then
362         result = { status = 99, error = "out of memory"}
363     else
364         result = mpx:execute(preamble)
365     end
366     log = reporterror(result)
367     return mpx, result, log
368 end

```

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

```

369 local process_stack = 0
370 function process (data, instancename)
371     local currfmt
372     process_stack = process_stack + 1
373     if instancename and instancename ~= "" then
374         currfmt = instancename
375         has_instancename = true
376     else
377         currfmt = tableconcat{
378             currentformat,
379             luamplib.numbersystem or "scaled",
380             tostring(luamplib.texttextlabel),
381             tostring(luamplib.legacyverbatimtex),
382             tostring(process_stack), -- try to address #63
383         }
384         has_instancename = false
385     end
386     local mpx = mplibinstances[currfmt]
387     local standalone = not (has_instancename or luamplib.codeinherit)
388     if mpx and standalone then
389         mpx:finish()
390     end
391     local log = ""
392     if standalone or not mpx then
393         mpx, _, log = luamplibload(currentformat)
394         mplibinstances[currfmt] = mpx
395     end
396     local converted, result = false, {}
397     if mpx and data then
398         result = mpx:execute(data)
399         local log = reporterror(result, log)
400         if log then
401             if result.fig then
402                 converted = luamplib.convert(result)
403             end
404         end
405     else
406         err"Mem file unloadable. Maybe generated with a different version of mplib?"
407     end
408     process_stack = process_stack - 1

```

```

409     return converted, result
410 end
411 end
412

```

dvipdfmx is supported, though nobody seems to use it.

```

413 local pdfmode = tex.outputmode > 0
414

```

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

```

415 local catlatex = luatexbase.registernumber("catcodetable@latex")
416 local catat11 = luatexbase.registernumber("catcodetable@atletter")

```

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

```

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

```

For conversion of sp to bp.

```

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

```

Prepare 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.

```

422 local texboxes = { globalid = 0, localid = 4096 }
423 local process_tex_text
424 do
425   local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
426     xscaled %f yscaled %f shifted (0,-%f) \z
427     withprescript "mplibtexboxid=%i:%f:%f")'
428   function process_tex_text (str, maketext)
429     if str then
430       if not maketext then str = str:gsub("\r.-$", "") end
431       local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
432         and "\\global" or ""
433       local tex_box_id
434       if global == "" then
435         tex_box_id = texboxes.localid + 1
436         texboxes.localid = tex_box_id
437       else
438         local boxid = texboxes.globalid + 1
439         texboxes.globalid = boxid
440         run_tex_code(format([[\\expandafter\\newbox\\csname luamplib.box.%s\\endcsname]], boxid))
441         tex_box_id = tex.getcount'allocationnumber'
442       end
443       if str:find"^[taggingoff%]" then
444         str = str:gsub("^[taggingoff%]*s*", "")
445         run_tex_code(format("\\luamplibnotagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",

```

```

446             tex_box_id, global, tex_box_id, str))
447     else
448         run_tex_code(format("\\luamplibtagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
449             tex_box_id, global, tex_box_id, str))
450     end
451     local box = texgetbox(tex_box_id)
452     local wd = box.width / factor
453     local ht = box.height / factor
454     local dp = box.depth / factor
455     return texttext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
456 end
457 return ""
458 end
459 end
460

```

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

```

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

```

A common function for color functions

```

475 local function is_xcolor (str)
476     for _,v in ipairs(str:explode"!") do
477         if not v:find("^%s*%d+%s*$") and is_defined("\\color@".v) then -- priority to xcolor
478             return true
479         end
480     end
481     return false
482 end
483 local function colorsplit (res)
484     local t, tt = { }, res:explode()
485     local be = tt[1]:find"^%d" and 1 or 2
486     for i=be, #tt do
487         if not tonumber(tt[i]) then break end
488         t[#t+1] = tt[i]
489     end
490     return t

```

```

491 end
492 do
493   local colfmt = ccexplat and "l3color" or "xcolor"
494   local mplibcolorfmt = {

```

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

```

495   xcolor = [[{\setbox0\hbox{{\color%s\global\mplibtmptoks\expanded{{\current@color}}}}}],
496   l3color = [[\color_export:nnN%s{raw}\l_tmpa_tl\mplibtmptoks\expandafter{\l_tmpa_tl}]]
497   }
498   function process_color (str)
499     if str then
500       if not str:find("%b{") then
501         str = format("%s",str)
502       end
503       local myfmt = mplibcolorfmt[colfmt]
504       if colfmt == "l3color" and is_defined"color" then
505         if str:find("%b[") or is_xcolor(str:match"{{(.+)}}") then
506           myfmt = mplibcolorfmt.xcolor
507         end
508       end
509       run_tex_code(myfmt:format(str), ccexplat or catat11)
510       local t = texgettoks"mplibtmptoks"
511       if not pdfmode then
512         if t:find"hsb" then
513           local spec = t:gsub("^hsb ", ""):gsub(" ", ",")
514           run_tex_code{"\\convertcolorspec{hsb}{", spec, "}{{rgb}}\\mplibtmpa"}
515           t = format("rgb %s", get_macro"mplibtmpa":gsub(","," "))
516         elseif not t:find"d" then -- named color
517           if not is_defined"extractcolorspecs" then err"needs xcolor package loaded" end
518           run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
519           t = format("%s %s", get_macro"mplibtmpa", get_macro"mplibtmpb":gsub(","," "))
520         end
521         local name, value = t:match"^(%a+) (.+)"
522         if name and value then
523           local op = name == "rgb" and "rg" or name == "cmyk" and "k" or name == "gray" and "g"
524           if not op then err"unknown color model" end
525           t = ("%s %s %s %s"):format(value, op, value, op:upper())
526         end
527         elseif is_defined"ver@colorspace.sty" and t:find"/&" then
528           local a,b,c,d = t:match"^(.- cs) (.- CS) (.- scn?) (.- SCN?)$"
529           if a and b and c and d then
530             t = tableconcat({ a, c, b, d }, " ")
531           end
532         end
533         return format('1 withprescript "mpliboverridecolor=%s"', t)
534       end
535       return ""
536     end

```

```

537 function process_mplibcolor(str)
538   local res = process_color(str)
539   if res:find" cs " then return res end
540   res = res:match'"mpliboverridecolor=(.)"'
541   local t = colorsplit(res)
542   if #t == 0 then
543     err("Color processing failed '%s'. It could be a luamplib bug. Please report.", res)
544   end
545   return format("(%s)", tableconcat(t, ","))
546 end
547 end
548
   for \mpdim or mplibdimen
549 local function process_dimen (str)
550   if str then
551     str = str:gsub("{(.+)}", "%1")
552     run_tex_code(format([[ \mplibtmp toks \expandafter { \the \dimexpr %s \relax } ]], str))
553     return format("begin group %s end group", texgettoks "mplibtmp toks")
554   end
555   return ""
556 end
557

```

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

```

558 local function process_verbatimtex_text (str)
559   if str then
560     run_tex_code(str)
561   end
562   return ""
563 end
564

```

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.

```

565 local tex_code_pre_mplib = {}
566 luamplib.figid = 1
567 luamplib.in_the_fig = false
568 local function process_verbatimtex_prefig (str)
569   if str then
570     tex_code_pre_mplib[luamplib.figid] = str
571   end
572   return ""
573 end
574 local function process_verbatimtex_infig (str)
575   if str then
576     return format('special "postmplibverbtex=%s";', str)
577   end
578   return ""
579 end

```

580

For *metafun* format. see issue #79.

```
581 mp = mp or {}
582 local mp = mp
583 mp.mf_path_reset = mp.mf_path_reset or function() end
584 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
585 mp.report = mp.report or info
```

metafun 2021-03-09 changes crashes luamplib.

```
586 catcodes = catcodes or {}
587 local catcodes = catcodes
588 catcodes.numbers = catcodes.numbers or {}
589 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
590 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
591 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
592 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
593 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
594 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
595 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
596
```

Now luamplib.runscript

```
597 do
598   local runscript_funcs = {
599     luamplibtext    = process_tex_text,
600     luamplibcolor   = process_mplibcolor,
601     luamplibdimen   = process_dimen,
602     luamplibprefig   = process_verbatimt看_text_prefig,
603     luamplibinfig    = process_verbatimt看_text_infig,
604     luamplibverbtex  = process_verbatimt看_text_text,
605   }
```

A function from ConT_EXt general.

```
606 local function mpprint(buffer,...)
607   for i=1,select("#",...) do
608     local value = select(i,...)
609     if value ~= nil then
610       local t = type(value)
611       if t == "number" then
612         buffer[#buffer+1] = format("%.16f",value)
613       elseif t == "string" then
614         buffer[#buffer+1] = value
615       elseif t == "table" then
616         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
617       else -- boolean or whatever
618         buffer[#buffer+1] = tostring(value)
619       end
620     end
621   end
622 end
623 function luamplib.runscript (code)
```

```

624 local id, str = code:match("(.-){(.*)}")
625 if id and str then
626     local f = runscript_funcs[id]
627     if f then
628         local t = f(str)
629         if t then return t end
630     end
631 end
632 local f = load(code)
633 if type(f) == "function" then
634     local buffer = {}
635     function mp.print(...)
636         mpprint(buffer,...)
637     end
638     local res = {f()}
639     buffer = tableconcat(buffer)
640     if buffer and buffer ~= "" then
641         return buffer
642     end
643     buffer = {}
644     mpprint(buffer, tableunpack(res))
645     return tableconcat(buffer)
646 end
647 return ""
648 end
649 end
650

```

luamplib.maketext

```

651 luamplib.legacyverbatimmtex = true
652 do

```

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

```

653 local function protecttexcontents (str)
654     return str:gsub("\\%", "\\0PerCent\0")
655         :gsub("%%.-\n", "")
656         :gsub("%%.-$", "")
657         :gsub("\\0PerCent\0", "\\%")
658         :gsub("\\r.-$", "")
659         :gsub("%s+", " ")
660 end
661 function luamplib.maketext (str, what)
662     if str and str ~= "" then
663         str = protecttexcontents(str)
664         if what == 1 then
665             if not str:find("\\documentclass"..name_e) and
666                 not str:find("\\begin%s*{document}") and
667                 not str:find("\\documentstyle"..name_e) and
668                 not str:find("\\usepackage"..name_e) then
669                 if luamplib.legacyverbatimmtex then

```

```

670         if luamplib.in_the_fig then
671             return process_verbatimtex_infig(str)
672         else
673             return process_verbatimtex_prefig(str)
674         end
675     else
676         return process_verbatimtex_text(str)
677     end
678 end
679 else
680     return process_tex_text(str, true) -- bool is for 'char13'
681 end
682 end
683 return ""
684 end
685 end
686
    luamplib's METAPOST color operators
687 local function get_spc_name_obj (spot)
688     if is_defined"spc@csall" then
689         for v in get_macro"spc@csall":gmatch"{(.-)}" do
690             local n, r = get_macro("spc@ir@".v):match"^(.-) (%d+ 0 R)"
691             if spot == n then
692                 return v, r
693             end
694         end
695     else
696         err"only l3color or colorspace is supported for spot color"
697     end
698 end
699 do
700     local function cmyk2rgb (t)
701         return {
702             1 - math.min(1, t[1]+t[4]),
703             1 - math.min(1, t[2]+t[4]),
704             1 - math.min(1, t[3]+t[4]),
705         }
706     end
707     luamplib.gettexcolor = function (str, rgb)
708         local res = process_color(str):match"mpliboverridecolor=(.+)""
709         if res:find" cs " then
710             info("%s is a spot color. Forced to %s", str, rgb and "RGB" or "CMYK")
711             if not is_xcolor(str) then
712                 run_tex_code({
713                     "\\color_export:nnN{" ,
714                     str,
715                     "}{" ,
716                     rgb and "space-sep-rgb" or "space-sep-cmyk",
717                     "}\\mplib@tempa",

```

```

718     },ccexplat)
719     return get_macro"mplib@tempa":explode()
720   else
721     local name, value = res:match"^(.-) cs (.-) sc"
722     local t = get_macro("spc@ascmyk@"..get_spc_name_obj(name)):explode"," -- mixed not work
723     for i = 1, #t do
724       t[i] = t[i] * value
725     end
726     return rgb and cmyk2rgb(t) or t
727   end
728 end
729 local t = colorsplit(res)
730 if #t == 3 or not rgb then return t end
731 if #t == 4 then return cmyk2rgb(t) end
732 return { t[1], t[1], t[1] }
733 end
734 end
735 luamplib.shadecolor = function (str)
736   local res = process_color(str):match'"mpliboverridecolor=(.+)"'
737   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}

738   local name, value, obj
739   if not is_xcolor(str) then
740     run_tex_code({ "\color_export:nnN{", str, "}{backend}\mplib@tempa" }, ccexplat)
741     name, value = get_macro'mplib@tempa':match'{{(.-)}{{(.-)}}}'
742     local t = res:explode()
743     local refname = t[1]:sub(2,-1)
744     if pdfmode then
745       obj = format("%s 0 R", ltx.pdf.object_id(refname))

```

```

746     else
747         run_tex_code({ "\\mplibtmp toks\\expanded{{{\\pdf_object_ref:n{", refname, "}}}" }, ccexplat)
748         obj = texgettoks"mplibtmp toks"
749     end
750     else -- colorspace: mixing is allowed only in preamble
751         name, value = res:match"^(.-) cs (.-) sc"
752         name, obj = get_spc_name_obj(name)
753     end
754     return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
755 end
756 return format('(%s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),","), str)
757 end
758

```

luamplib.fillandstrokecolor

```

759 do
760     local function graphictextcolor (col, filldraw)
761         if col:find"^[%d%.:]+$" then
762             col = col:explode":"
763             for i=1,#col do
764                 col[i] = format("%.3f", col[i])
765             end
766             local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
767             col[#col+1] = filldraw == "fill" and op or op:upper()
768             return tableconcat(col," ")
769         end
770         col = process_color(col):match'"mpliboverridecolor=(.+)"'
771         local t = col:explode()
772         local b = filldraw == "fill" and 1 or #t/2+1
773         local e = b == 1 and #t/2 or #t
774         return tableconcat(t," ", b, e)
775     end
776     function luamplib.fillandstrokecolor (fill, stroke)
777         fill = graphictextcolor(fill, "fill")
778         stroke = graphictextcolor(stroke, "stroke")
779         return format('withprescript "mpliboverridecolor=%s %s"', fill, stroke)
780     end
781 end
782

```

Remove trailing zeros for smaller PDF

```

783 local decimals = "%.d+"
784 local function rmzeros(str) return str:gsub("%.?0+$", "") end
785

```

common function for mplibgraphictext and mpliboutlinetext

```

786 local function getrulemetric (box, curr, bp)
787     local running = -1073741824
788     local wd,ht,dp = curr.width, curr.height, curr.depth
789     wd = wd == running and box.width or wd

```

```

790 ht = ht == running and box.height or ht
791 dp = dp == running and box.depth or dp
792 if bp then
793   return wd/factor, ht/factor, dp/factor
794 end
795 return wd, ht, dp
796 end
797

```

luamplib's mplibgraphicstext operator

```

798 do
799   if not math.round then
800     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
801   end
802   local emboldenfonts = { }
803   local function roundupwidth (f, fb)
804     local wd = math.round(f.size * fb / factor * 10)
805     if wd == 0 and fb ~= 0 then
806       wd = 1
807     end
808     emboldenfonts.width = wd
809     return wd
810   end
811   local function getemboldenwidth (curr, fakebold)
812     local width = emboldenfonts.width
813     if not width then
814       local f
815       local function getglyph(n)
816         while n do
817           if n.head then
818             getglyph(n.head)
819           elseif n.font and n.font > 0 then
820             f = n.font; break
821           end
822           n = node.getnext(n)
823         end
824       end
825       getglyph(curr)
826       width = roundupwidth(font.getcopy(f or font.current()), fakebold)
827     end
828     return width
829   end
830   local function getrulewhatsit (line, wd, ht, dp)
831     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
832     line = line == 0 and "" or ("%f w"):format(line)
833     local pl
834     local fmt = "q %s %f %f %f %f re B Q"
835     if pdfmode then
836       pl = node.new("whatsit", "pdf_literal")

```

```

837     pl.mode = 0
838   else
839     fmt = "pdf:content " .. fmt
840     pl = node.new("whatsit", "special")
841   end
842   pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
843   local ss = node.new"glue"
844   node.setglue(ss, 0, 65536, 65536, 2, 2)
845   pl.next = ss
846   return pl
847 end

```

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

```

848 local tag_update_attrs
849 if is_defined"ver@tagpdf.sty" then
850   tag_update_attrs = function (n, curr)
851     while n do
852       n.attr = curr.attr
853       if n.head then
854         tag_update_attrs(n.head, curr)
855       end
856       n = node.getnext(n)
857     end
858   end
859 else
860   tag_update_attrs = function() end
861 end
862 local function embolden (box, curr, fakebold)
863   local head = curr
864   while curr do
865     if curr.head then
866       curr.head = embolden(curr, curr.head, fakebold)
867     elseif curr.replace then
868       curr.replace = embolden(box, curr.replace, fakebold)
869     elseif curr.leader then
870       if curr.leader.head then
871         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
872       elseif curr.leader.id == node.id"rule" then
873         local glue = node.effective_glue(curr, box)
874         local line = getemboldenwidth(curr, fakebold)
875         local wd, ht, dp = getrulemetric(box, curr.leader)
876         if box.id == node.id"hlist" then
877           wd = glue
878         else
879           ht, dp = 0, glue
880         end
881         local pl = getrulewhatsit(line, wd, ht, dp)
882         local pack = box.id == node.id"hlist" and node.hpack or node.vpack
883         local list = pack(pl, glue, "exactly")

```

```

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

```

```

933     local fmt = process_tex_text(text):sub(1,-2)
934     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
935     emboldenfonts.width = nil
936     local box = texgetbox(id)
937     box.head = embolden(box, box.head, fakebold)
938     local colors = luamplib.fillandstrokecolor(fc, dc)
939     return format('%s %s)', fmt, colors)
940 end
941 end
942

```

luamplib's mplibglyph operator

```

943 do
944     local function mperr (str)
945         return format("hide(errmessage %q)", str)
946     end
947     local function getangle (a,b,c)
948         local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
949         if r > 180 then
950             r = r - 360
951         elseif r < -180 then
952             r = r + 360
953         end
954         return r
955     end
956     local function turning (t)
957         local r, n = 0, #t
958         for i=1,2 do
959             tableinsert(t, t[i])
960         end
961         for i=1,n do
962             r = r + getangle(t[i], t[i+1], t[i+2])
963         end
964         return r/360
965     end
966     local function glyphimage(t, fmt)
967         local q, p, r, towarn = {},{}
968         local function closepath(dots)
969             tableinsert(p, format("%scycle", dots or "--"))
970             tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
971         end
972         for i,v in ipairs(t) do
973             local cmd = v[#v]
974             local nt = t[i+1]
975             local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
976             if cmd == "m" then
977                 if final then towarn = true end
978                 p = {format('(%s,%s)',v[1],v[2])}
979                 r = {{x=v[1],y=v[2]}}

```

```

980     else
981         if cmd == "l" then
982             local pt = t[i-1]
983             if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
984                 else
985                     tableinsert(p, format('--(%s,%s)',v[1],v[2]))
986                     tableinsert(r, {x=v[1],y=v[2]})
987                 end
988                 if final then closepath() end
989             elseif cmd == "c" then
990                 tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
991                 if final and r[1].x == v[5] and r[1].y == v[6] then
992                     closepath ".."
993                 else
994                     tableinsert(p, format('..(%s,%s)',v[5],v[6]))
995                     tableinsert(r, {x=v[5],y=v[6]})
996                     if final then closepath() end
997                 end
998             elseif cmd == "path" or cmd == "move" then
999                 else
1000                     return mperr"unknown operator"
1001                 end
1002             end
1003         end
1004         r = { }
1005         if fmt == "opentype" then
1006             for _,v in ipairs(q[1]) do
1007                 tableinsert(r, format('addto currentpicture contour %s;',v))
1008             end
1009             for _,v in ipairs(q[2]) do
1010                 tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
1011             end
1012         else
1013             for _,v in ipairs(q[2]) do
1014                 tableinsert(r, format('addto currentpicture contour %s;',v))
1015             end
1016             for _,v in ipairs(q[1]) do
1017                 tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
1018             end
1019         end
1020         return format('image(%s)', tableconcat(r)), towarn
1021     end
1022 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
1023 function luamplib.glyph (f, c)
1024     local filename, subfont, instance, kind, shapedata
1025     local fid = tonumber(f) or font.id(f)
1026     if fid > 0 then
1027         local fontdata = font.getfont(fid) or font.getcopy(fid)
1028         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format

```

```

1029     instance = fontdata.specification and fontdata.specification.instance
1030     or fontdata.shared and fontdata.shared.features.axis
1031     filename = filename and filename:gsub("^harfloaded:", "")
1032 else
1033     local name
1034     f = f:match"^%s*(.)%s*$"
1035     name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
1036     if not name then
1037         name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
1038     end
1039     if not name then
1040         name, subfont = f:match"(.+)%((%d+)%)%" -- Times.ttc(2)
1041     end
1042     name = name or f
1043     subfont = (subfont or 0)+1
1044     instance = instance and instance:lower()
1045     for _,ftype in ipairs{"opentype", "truetype"} do
1046         filename = kpse.find_file(name, ftype.." fonts")
1047         if filename then
1048             kind = ftype; break
1049         end
1050     end
1051 end
1052 if kind ~= "opentype" and kind ~= "truetype" then
1053     f = fid and fid > 0 and tex.fontname(fid) or f
1054     if kpse.find_file(f, "tfm") then
1055         return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
1056     else
1057         filename = kpse.find_file(f, "type1 fonts")
1058         if filename then
1059             kind = "type1" -- there's bug in processing cmr family
1060         else
1061             return mperr"font not found"
1062         end
1063     end
1064 end
1065 local time = lfsattributes(filename,"modification")

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

1066 local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1067 local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
1068 local newname = format("%s/%s.lua", cachedir or outputdir(), h)
1069 local newtime = lfsattributes(newname,"modification") or 0
1070 if time == newtime then
1071     shapedata = require(newname)
1072 end
1073 if not shapedata then

```

```

1074     if fonts then
1075         local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1076         shapedata = handler.readers.loadshapes(filename, subfont, instance)
1077     end
1078     if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1079     table.tofile(newname, shapedata, "return")
1080     lfstouch(newname, time, time)
1081 end
1082 local gid = tonumber(c)
1083 if not gid then
1084     local uni = utf8.codepoint(c)
1085     for i,v in pairs(shapedata.glyphs) do
1086         if c == v.name or uni == v.unicode then
1087             gid = i; break
1088         end
1089     end
1090 end
1091 if not gid then return mperr"cannot get GID (glyph id)" end
1092 local fac = 1000 / (shapedata.units or 1000)
1093 local t = shapedata.glyphs[gid]; t = t and t.segments
1094 if not t then return "image()" end
1095 for i,v in ipairs(t) do
1096     if type(v) == "table" then
1097         for ii,vv in ipairs(v) do
1098             if type(vv) == "number" then
1099                 t[i][ii] = format("%.0f", vv * fac)
1100             end
1101         end
1102     end
1103 end
1104 local result, towarn = glyphimage(t, shapedata.format or kind)
1105 if towarn then
1106     warn("mplibglyph %s not working properly. Use glyph instead", f)
1107 end
1108 return result
1109 end
1110 end
1111

```

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

```

1112 do
1113     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1114         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1115     local outline_horz, outline_vert
1116     function outline_vert (res, box, curr, xshift, yshift)
1117         local b2u = box.dir == "LTL"
1118         local dy = (b2u and -box.depth or box.height)/factor
1119         local ody = dy
1120         while curr do

```

```

1121 if curr.id == node.id"rule" then
1122     local wd, ht, dp = getrulemetric(box, curr, true)
1123     local hd = ht + dp
1124     if hd ~= 0 then
1125         dy = dy + (b2u and dp or -ht)
1126         if wd ~= 0 and curr.subtype == 0 then
1127             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1128         end
1129         dy = dy + (b2u and ht or -dp)
1130     end
1131 elseif curr.id == node.id"glue" then
1132     local vwidth = node.effective_glue(curr,box)/factor
1133     if curr.leader then
1134         local curr, kind = curr.leader, curr.subtype
1135         if curr.id == node.id"rule" then
1136             local wd = getrulemetric(box, curr, true)
1137             if wd ~= 0 then
1138                 local hd = vwidth
1139                 local dy = dy + (b2u and 0 or -hd)
1140                 if hd ~= 0 and curr.subtype == 0 then
1141                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1142                 end
1143             end
1144         elseif curr.head then
1145             local hd = (curr.height + curr.depth)/factor
1146             if hd <= vwidth then
1147                 local dy, n, iy = dy, 0, 0
1148                 if kind == 100 or kind == 103 then -- todo: gleaders
1149                     local ady = abs(ody - dy)
1150                     local ndy = math.ceil(ady / hd) * hd
1151                     local diff = ndy - ady
1152                     n = math.floor((vwidth-diff) / hd)
1153                     dy = dy + (b2u and diff or -diff)
1154                 else
1155                     n = math.floor(vwidth / hd)
1156                     if kind == 101 then
1157                         local side = vwidth % hd / 2
1158                         dy = dy + (b2u and side or -side)
1159                     elseif kind == 102 then
1160                         iy = vwidth % hd / (n+1)
1161                         dy = dy + (b2u and iy or -iy)
1162                     end
1163                 end
1164                 dy = dy + (b2u and curr.depth or -curr.height)/factor
1165                 hd = b2u and hd or -hd
1166                 iy = b2u and iy or -iy
1167                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1168                 for i=1,n do
1169                     res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)

```

```

1170         dy = dy + hd + iy
1171     end
1172 end
1173 end
1174 end
1175 dy = dy + (b2u and vwidth or -vwidth)
1176 elseif curr.id == node.id"kern" then
1177     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1178 elseif curr.id == node.id"vlist" then
1179     dy = dy + (b2u and curr.depth or -curr.height)/factor
1180     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1181     dy = dy + (b2u and curr.height or -curr.depth)/factor
1182 elseif curr.id == node.id"hlist" then
1183     dy = dy + (b2u and curr.depth or -curr.height)/factor
1184     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1185     dy = dy + (b2u and curr.height or -curr.depth)/factor
1186 end
1187 curr = node.getnext(curr)
1188 end
1189 return res
1190 end
1191 function outline_horz (res, box, curr, xshift, yshift, discwd)
1192     local r2l = box.dir == "TRT"
1193     local dx = r2l and (discwd or box.width/factor) or 0
1194     local dirs = { { dir = r2l, dx = dx } }
1195     while curr do
1196         if curr.id == node.id"dir" then
1197             local sign, dir = curr.dir:match"(.)(...)"
1198             local level, newdir = curr.level, r2l
1199             if sign == "+" then
1200                 newdir = dir == "TRT"
1201                 if r2l ~= newdir then
1202                     local n = node.getnext(curr)
1203                     while n do
1204                         if n.id == node.id"dir" and n.level+1 == level then break end
1205                         n = node.getnext(n)
1206                     end
1207                     n = n or node.tail(curr)
1208                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1209                 end
1210                 dirs[level] = { dir = r2l, dx = dx }
1211             else
1212                 local level = level + 1
1213                 newdir = dirs[level].dir
1214                 if r2l ~= newdir then
1215                     dx = dirs[level].dx
1216                 end
1217             end
1218             r2l = newdir

```

```

1219 elseif curr.char and curr.font and curr.font > 0 then
1220     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1221     local gid = ft.characters[curr.char].index or curr.char
1222     local scale = ft.size / factor / 1000
1223     local slant = (ft.slant or 0)/1000
1224     local extend = (ft.extend or 1000)/1000
1225     local squeeze = (ft.squeeze or 1000)/1000
1226     local expand = 1 + (curr.expansion_factor or 0)/1000000
1227     local xscale, yscale = scale * extend * expand, scale * squeeze
1228     dx = dx - (r2l and curr.width/factor*expand or 0)
1229     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1230     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1231     local vertical = ""
1232     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1233         if ft.shared.features.vertical then -- luatexko
1234             vertical = "rotated 90"
1235             local data = ft.characters[curr.char] or { }
1236             if ft.hb then
1237                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1238                 local charraise = (ft.luatexko_charraise or 0)/factor
1239                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1240             else
1241                 local cmds = data.commands or { {0,0}, {0,0} }
1242                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1243                 xpos, ypos = xpos + hoff, ypos + voff
1244             end
1245         elseif curr ~= box.head then -- luatexja
1246             vertical = "rotated 90"
1247             local en = ft.parameters.quad/factor/2
1248             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1249         end
1250     end
1251     local image
1252     if ft.format == "opentype" or ft.format == "truetype" then
1253         image = luamplib.glyph(curr.font, gid)
1254     else
1255         local name, scale = ft.name, 1
1256         local vf = font.read_vf(name, ft.size)
1257         if vf and vf.characters[gid] then
1258             local cmds = vf.characters[gid].commands or { }
1259             for _,v in ipairs(cmds) do
1260                 if v[1] == "char" then
1261                     gid = v[2]
1262                 elseif v[1] == "font" and vf.fonts[v[2]] then
1263                     name = vf.fonts[v[2]].name
1264                     scale = vf.fonts[v[2]].size / ft.size
1265                 end
1266             end
1267         end
1268     end

```

```

1268         image = format("glyph %s of %q scaled %f", gid, name, scale)
1269     end
1270     res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1271                          #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1272     dx = dx + (r2l and 0 or curr.width/factor*expand)
1273 elseif curr.replace then
1274     local width = node.dimensions(curr.replace)/factor
1275     dx = dx - (r2l and width or 0)
1276     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1277     dx = dx + (r2l and 0 or width)
1278 elseif curr.id == node.id"rule" then
1279     local wd, ht, dp = getrulemetric(box, curr, true)
1280     if wd ~= 0 then
1281         local hd = ht + dp
1282         dx = dx - (r2l and wd or 0)
1283         if hd ~= 0 and curr.subtype == 0 then
1284             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1285         end
1286         dx = dx + (r2l and 0 or wd)
1287     end
1288 elseif curr.id == node.id"glue" then
1289     local width = node.effective_glue(curr, box)/factor
1290     dx = dx - (r2l and width or 0)
1291     if curr.leader then
1292         local curr, kind = curr.leader, curr.subtype
1293         if curr.id == node.id"rule" then
1294             local wd, ht, dp = getrulemetric(box, curr, true)
1295             local hd = ht + dp
1296             if hd ~= 0 then
1297                 wd = width
1298                 if wd ~= 0 and curr.subtype == 0 then
1299                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1300                 end
1301             end
1302         end
1303     elseif curr.head then
1304         local wd = curr.width/factor
1305         if wd <= width then
1306             local dx = r2l and dx+width or dx
1307             local n, ix = 0, 0
1308             if kind == 100 or kind == 103 then -- todo: gleaders
1309                 local adx = abs(dx-dirs[1].dx)
1310                 local ndx = math.ceil(adx / wd) * wd
1311                 local diff = ndx - adx
1312                 n = math.floor((width-diff) / wd)
1313                 dx = dx + (r2l and -diff-wd or diff)
1314             else
1315                 n = math.floor(width / wd)
1316                 if kind == 101 then
1317                     local side = width % wd / 2

```

```

1317         dx = dx + (r2l and -side-wd or side)
1318     elseif kind == 102 then
1319         ix = width % wd / (n+1)
1320         dx = dx + (r2l and -ix-wd or ix)
1321     end
1322 end
1323 wd = r2l and -wd or wd
1324 ix = r2l and -ix or ix
1325 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1326 for i=1,n do
1327     res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1328     dx = dx + wd + ix
1329 end
1330 end
1331 end
1332 end
1333 dx = dx + (r2l and 0 or width)
1334 elseif curr.id == node.id"kern" then
1335     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1336 elseif curr.id == node.id"math" then
1337     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1338 elseif curr.id == node.id"vlist" then
1339     dx = dx - (r2l and curr.width/factor or 0)
1340     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1341     dx = dx + (r2l and 0 or curr.width/factor)
1342 elseif curr.id == node.id"hlist" then
1343     dx = dx - (r2l and curr.width/factor or 0)
1344     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1345     dx = dx + (r2l and 0 or curr.width/factor)
1346 end
1347 curr = node.getnext(curr)
1348 end
1349 return res
1350 end
1351 function luamplib.outlinetext (text)
1352     local fmt = process_tex_text(text)
1353     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1354     local box = texgetbox(id)
1355     local res = outline_horz({ }, box, box.head, 0, 0)
1356     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1357     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1358 end
1359 end
1360

```

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

```

1361 function luamplib.getunicodegraphemes (s)
1362     local t = { }
1363     local graphemes = require'lua-uni-graphemes'

```

```

1364 for _, _, c in graphemes.graphemes(s) do
1365   table.insert(t, c)
1366 end
1367 return t
1368 end
1369 function luamplib.unicodesubstring (s,b,e,grph)
1370   local tt, t, step = { }
1371   if grph then
1372     t = luamplib.getunicodegraphemes(s)
1373   else
1374     t = { }
1375     for _, c in utf8.codes(s) do
1376       table.insert(t, utf8.char(c))
1377     end
1378   end
1379   if b <= e then
1380     b, step = b+1, 1
1381   else
1382     e, step = e+1, -1
1383   end
1384   for i = b, e, step do
1385     table.insert(tt, t[i])
1386   end
1387   s = table.concat(tt):gsub("'", "'&ditto'")
1388   return string.format("%s", s)
1389 end
1390

```

METAPOST preambles

```

1391 luamplib.preambles = {
1392   preamble = [[
1393     boolean mplib ; mplib := true ;
1394     let dump = endinput ;
1395     let normalfontsize = fontsize;
1396     input %s ;
1397   ]],
1398   mplibcode = [[
1399     texscriptmode := 2;
1400     def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1401     def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1402     def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1403     def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1404     if known context_mlib:
1405       defaultfont := "cmtt10";
1406       let infont = normalinfont;
1407       let fontsize = normalfontsize;
1408       vardef thelabel@#(expr p,z) =
1409         if string p :
1410           thelabel@#(p infont defaultfont scaled defaultscale,z)

```

```

1411     else :
1412         p shifted (z + labeloffset*mfun_laboff@# -
1413             (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1414             (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1415     fi
1416 enddef;
1417 else:
1418     vardef texttext@# primary t = rawtexttext (t) enddef;
1419     def message expr t =
1420         if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1421     enddef;
1422     def withtransparency (expr a, t) =
1423         withprescript "tr_alternative=" & if numeric a: decimal fi a
1424         withprescript "tr_transparency=" & decimal t
1425     enddef;
1426     vardef ddecimal primary p =
1427         decimal xpart p & " " & decimal ypart p
1428     enddef;
1429     vardef boundingbox primary p =
1430         if (path p) or (picture p) :
1431             llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1432         else :
1433             origin
1434         fi -- cycle
1435     enddef;
1436 fi
1437 def resolvedcolor(expr s) =
1438     runscript("return luamplib.shadecolor('"&s&"')")
1439 enddef;
1440 def colordecimals primary c =
1441     if cmykcolor c:
1442         decimal cyanpart c & ":" & decimal magentapart c & ":" &
1443         decimal yellowpart c & ":" & decimal blackpart c
1444     elseif rgbcolor c:
1445         decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1446     elseif string c:
1447         if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1448     else:
1449         decimal c
1450     fi
1451 enddef;
1452 def externalfigure primary filename =
1453     draw rawtexttext("\includegraphics{"&filename &}")
1454 enddef;
1455 def TEX = texttext enddef;
1456 def mplibtexcolor primary c =
1457     runscript("return luamplib.gettexcolor('"&c&"')")
1458 enddef;
1459 def mplibrgbtexcolor primary c =

```

```

1460 runscript("return luamplib.gettexcolor('& c &''', 'rgb')")
1461 enddef;
1462 def mplibgraphictext primary t =
1463   begingroup;
1464   mplibgraphictext_ (t)
1465 enddef;
1466 def mplibgraphictext_ (expr t) text rest =
1467   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1468   fb, fc, dc, graphictextpic, alsoordoublepath;
1469   picture graphictextpic; graphictextpic := nullpicture;
1470   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1471   let scale = scaled;
1472   def fakebold primary c = hide(fb:=c;) enddef;
1473   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1474   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1475   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1476   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1477   addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1478   def fakebold primary c = enddef;
1479   let fillcolor = fakebold; let drawcolor = fakebold;
1480   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1481   image(draw runscript("return luamplib.graphictext([===['&t&']===], "
1482     & decimal fb &'', '& fc &''', '& dc &''')") rest;)
1483   endgroup;
1484 enddef;
1485 def mplibglyph expr c of f =
1486   runscript (
1487     "return luamplib.glyph('"
1488     & if numeric f: decimal fi f
1489     & " ', '"
1490     & if numeric c: decimal fi c
1491     & " ')"
1492   )
1493 enddef;
1494 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1495 def mplibdrawglyph expr g =
1496   luamplib_tmp_num_ := 0;
1497   for item within g:
1498     fill pathpart item
1499     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1500   endfor
1501 enddef;
1502 let mplibfillglyph = mplibdrawglyph;
1503 def mplibstrokeyglyph expr g =
1504   luamplib_tmp_num_ := 0;
1505   for item within g:
1506     draw pathpart item
1507     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1508   endfor

```

```

1509 enddef;
1510 def mplibfillandstrokeglyph expr g =
1511   luamplib_tmp_num_ := 0;
1512   for item within g:
1513     draw pathpart item withpostscript
1514     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1515   endfor
1516 enddef;
1517 def withmplibcolors (expr f, s) =
1518   runscript("return luamplib.fillandstrokecolor(' " &
1519     if not string f: colordecimals fi f & "''," &
1520     if not string s: colordecimals fi s & "'')")
1521 enddef;
1522 def withmplibopacities (expr a, f, s) =
1523   withprescript "tr_alternative=" & if numeric a: decimal fi a
1524   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1525 enddef;
1526 def mplib_do_outline_text_set_b (text f) (text d) text r =
1527   def mplib_do_outline_options_f = f enddef;
1528   def mplib_do_outline_options_d = d enddef;
1529   def mplib_do_outline_options_r = r enddef;
1530 enddef;
1531 def mplib_do_outline_text_set_f (text f) text r =
1532   def mplib_do_outline_options_f = f enddef;
1533   def mplib_do_outline_options_r = r enddef;
1534 enddef;
1535 def mplib_do_outline_text_set_u (text f) text r =
1536   def mplib_do_outline_options_f = f enddef;
1537 enddef;
1538 def mplib_do_outline_text_set_d (text d) text r =
1539   def mplib_do_outline_options_d = d enddef;
1540   def mplib_do_outline_options_r = r enddef;
1541 enddef;
1542 def mplib_do_outline_text_set_r (text d) (text f) text r =
1543   def mplib_do_outline_options_d = d enddef;
1544   def mplib_do_outline_options_f = f enddef;
1545   def mplib_do_outline_options_r = r enddef;
1546 enddef;
1547 def mplib_do_outline_text_set_n text r =
1548   def mplib_do_outline_options_r = r enddef;
1549 enddef;
1550 def mplib_do_outline_text_set_p = enddef;
1551 def mplib_fill_outline_text =
1552   for n=1 upto mpliboutlinenum:
1553     i:=0;
1554     for item within mpliboutlinepic[n]:
1555       i:=i+1;
1556       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1557       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi

```

```

1558   endfor
1559 endfor
1560 enddef;
1561 def mplib_draw_outline_text =
1562   for n=1 upto mpliboutlinenum:
1563     for item within mpliboutlinepic[n]:
1564       draw pathpart item mplib_do_outline_options_d;
1565     endfor
1566   endfor
1567 enddef;
1568 def mplib_filldraw_outline_text =
1569   for n=1 upto mpliboutlinenum:
1570     i:=0;
1571     for item within mpliboutlinepic[n]:
1572       i:=i+1;
1573       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1574         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1575       else:
1576         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1577       fi
1578     endfor
1579   endfor
1580 enddef;
1581 vardef mpliboutlinetext@# (expr t) text rest =
1582   save kind; string kind; kind := str @#;
1583   save i; numeric i;
1584   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1585   def mplib_do_outline_options_d = enddef;
1586   def mplib_do_outline_options_f = enddef;
1587   def mplib_do_outline_options_r = enddef;
1588   runscript("return luamplib.outlinetext[==["&t&"]==]");
1589   image ( addto currentpicture also image (
1590     if kind = "f":
1591       mplib_do_outline_text_set_f rest;
1592       mplib_fill_outline_text;
1593     elseif kind = "d":
1594       mplib_do_outline_text_set_d rest;
1595       mplib_draw_outline_text;
1596     elseif kind = "b":
1597       mplib_do_outline_text_set_b rest;
1598       mplib_fill_outline_text;
1599       mplib_draw_outline_text;
1600     elseif kind = "u":
1601       mplib_do_outline_text_set_u rest;
1602       mplib_filldraw_outline_text;
1603     elseif kind = "r":
1604       mplib_do_outline_text_set_r rest;
1605       mplib_draw_outline_text;
1606       mplib_fill_outline_text;

```

```

1607     elseif kind = "p":
1608         mplib_do_outline_text_set_p;
1609         mplib_draw_outline_text;
1610     else:
1611         mplib_do_outline_text_set_n rest;
1612         mplib_fill_outline_text;
1613     fi;
1614 ) mplib_do_outline_options_r; )
1615 enddef ;
1616 def withmppattern primary p =
1617     withprescript "mplibpattern=" & if numeric p: decimal fi p
1618 enddef;
1619 def withpatternstroke expr a =
1620     withprescript "mplibpatternstroke=" & a
1621 enddef;
1622 primarydef t withpattern p =
1623     image(
1624         if cycle t:
1625             fill
1626         else:
1627             draw
1628         fi
1629         t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1630 enddef;
1631 vardef mplibtransformmatrix (text e) =
1632     save t; transform t;
1633     t = identity e;
1634     runscript("luamplib.transformmatrix = {"
1635     & decimal xpart t & ","
1636     & decimal ypart t & ","
1637     & decimal xpart t & ","
1638     & decimal ypart t & ","
1639     & decimal xpart t & ","
1640     & decimal ypart t & ","
1641     & "}");
1642 enddef;
1643 primarydef p withmaskinggroup s =
1644     if picture p:
1645         image(
1646             draw p;
1647             draw center p withprescript "mplibfadestate=stop";
1648         )
1649     else:
1650         p withprescript "mplibfadestate=stop"
1651     fi
1652     withprescript "mplibfadetype=masking"
1653     withprescript "mplibmaskname=" & s
1654 enddef;
1655 def withmaskingbgcolor expr c =

```

```

1656 withprescript "mplibmaskingbgcolor=" & colordecimals c
1657 enddef;
1658 primarydef p withfademethod s =
1659   if picture p:
1660     image(
1661       draw p;
1662       draw center p withprescript "mplibfadestate=stop";
1663     )
1664   else:
1665     p withprescript "mplibfadestate=stop"
1666   fi
1667   withprescript "mplibfadetype=" & s
1668   hide(mplib_shade_step := 1;)
1669   withprescript "sh_color_a=1"
1670   withprescript "sh_color_b=0"
1671   withprescript "mplibfadebbox=" &
1672     decimal (xpart llcorner p -1/4) & ":" &
1673     decimal (ypart llcorner p -1/4) & ":" &
1674     decimal (xpart urcorner p +1/4) & ":" &
1675     decimal (ypart urcorner p +1/4)
1676 enddef;
1677 def withfadevector (expr a,b) =
1678   withprescript "mplibfadevector=" &
1679     decimal xpart a & ":" &
1680     decimal ypart a & ":" &
1681     decimal xpart b & ":" &
1682     decimal ypart b
1683 enddef;
1684 let withfadecenter = withfadevector;
1685 def withfaderadius (expr a,b) =
1686   withprescript "mplibfaderadius=" &
1687     decimal a & ":" &
1688     decimal b
1689 enddef;
1690 def withfadebbox (expr a,b) =
1691   withprescript "mplibfadebbox=" &
1692     decimal xpart a & ":" &
1693     decimal ypart a & ":" &
1694     decimal xpart b & ":" &
1695     decimal ypart b
1696 enddef;
1697 def withadjustbbox expr n =
1698   withprescript "mplibadjustbbox=" & decimal n
1699 enddef;
1700 primarydef p asgroup s =
1701   image(
1702     draw center p
1703     withprescript "mplibgroupbbox=" &
1704     decimal (xpart llcorner p -1/4) & ":" &

```

```

1705     decimal (ypart llcorner p -1/4) & ":" &
1706     decimal (xpart urcorner p +1/4) & ":" &
1707     decimal (ypart urcorner p +1/4)
1708     withprescript "gr_state=start"
1709     withprescript "gr_type=" & s;
1710     draw p withprescript "sh_in_xobj=yes";
1711     draw center p withprescript "gr_state=stop";
1712 )
1713 enddef;
1714 def withgroupbbox (expr a,b) =
1715   withprescript "mplibgroupbbox=" &
1716   decimal xpart a & ":" &
1717   decimal ypart a & ":" &
1718   decimal xpart b & ":" &
1719   decimal ypart b
1720 enddef;
1721 def withgroupname expr s =
1722   withprescript "mplibgroupname=" & s
1723 enddef;
1724 def usemplibgroup primary s =
1725   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s & "\endcsname}")
1726   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1727 enddef;
1728 path    mplib_shade_path ;
1729 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1730 numeric mplib_shade_fx, mplib_shade_fy ;
1731 numeric mplib_shade_lx, mplib_shade_ly ;
1732 numeric mplib_shade_nx, mplib_shade_ny ;
1733 numeric mplib_shade_dx, mplib_shade_dy ;
1734 numeric mplib_shade_tx, mplib_shade_ty ;
1735 primarydef p withshadingmethod m =
1736   p
1737   if picture p :
1738     withprescript "sh_operand_type=picture"
1739     if textual p or (length p > 1):
1740       withprescript "sh_transform=no"
1741       mplib_with_shade_method (boundingbox p, m)
1742     else:
1743       withprescript "sh_transform=yes"
1744       mplib_with_shade_method (pathpart p, m)
1745     fi
1746   else :
1747     withprescript "sh_transform=yes"
1748     mplib_with_shade_method (p, m)
1749   fi
1750 enddef;
1751 def mplib_with_shade_method (expr p, m) =
1752   hide(mplib_with_shade_method_analyze(p))
1753   withprescript "sh_domain=0 1"

```

```

1754 withprescript "sh_color=into"
1755 withprescript "sh_color_a=" & colordecimals white
1756 withprescript "sh_color_b=" & colordecimals black
1757 withprescript "sh_first=" & ddecimal point 0 of p
1758 withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1759 withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1760 if m = "linear" :
1761   withprescript "sh_type=linear"
1762   withprescript "sh_factor=1"
1763   withprescript "sh_center_a=" & ddecimal llcorner p
1764   withprescript "sh_center_b=" & ddecimal urcorner p
1765 elseif m = "coons":
1766   withprescript "sh_type=coons"
1767   withprescript "sh_transform=no"
1768 elseif m = "triangle":
1769   withprescript "sh_type=triangle"
1770   withprescript "sh_transform=no"
1771 elseif m = "lattice":
1772   withprescript "sh_type=lattice"
1773   withprescript "sh_transform=no"
1774 elseif m = "tensor":
1775   withprescript "sh_type=tensor"
1776   withprescript "sh_transform=no"
1777   withprescript "sh_tensor_path=" &
1778     ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1779     ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1780     ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1781     ddecimal 1/4[point 3 of p, point 1 of p]
1782 else :
1783   withprescript "sh_type=circular"
1784   withprescript "sh_factor=1.2"
1785   withprescript "sh_center_a=" & ddecimal center p
1786   withprescript "sh_center_b=" & ddecimal center p
1787   withprescript "sh_radius_a=" & decimal 0
1788   withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1789 fi
1790 enddef;
1791 def withlatticeverticesperrow expr n =
1792   withprescript "sh_lattice_perrow=" & decimal n
1793 enddef;
1794 def withlatticeverticesdata (text t) =
1795   hide(luamplib_tmp_num_ := 0;)
1796   withprescript "sh_lattice_data=" &
1797   for item = t:
1798     if odd(incr luamplib_tmp_num_):
1799       if pair item: ddecimal fi item & " " &
1800     fi
1801   endfor ""
1802   hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)

```

```

1803 for item = t:
1804     if not odd(incr luamplib_tmp_num_):
1805         withshadingstep( withshadingcolors(item,item) )
1806     fi
1807 endfor
1808 enddef;
1809 def withtrianglepatchinit (expr p, a, b, c) =
1810     if string p:
1811         withtrianglepatchnext(0, p , a)
1812         withtrianglepatchnext(0, "", b)
1813         withtrianglepatchnext(0, "", c)
1814     else:
1815         withtrianglepatchnext(0, point 0 of p, a)
1816         withtrianglepatchnext(0, point 1 of p, b)
1817         withtrianglepatchnext(0, point 2 of p, c)
1818     fi
1819 enddef;
1820 def withtrianglepatchnext (expr f, p, a) =
1821     withshadingstep(
1822         withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1823         withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1824         if string p: p else: ddecimal p fi
1825         withshadingcolors (a, a)
1826     )
1827 enddef;
1828 def withcoonspatchinit (expr p, a, b, c, d) =
1829     withshadingstep(
1830         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1831         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1832         if string p: p
1833         else:
1834             ddecimal point      0 of p & " " &
1835             ddecimal postcontrol 0 of p & " " &
1836             ddecimal precontrol  1 of p & " " &
1837             ddecimal point      1 of p & " " &
1838             ddecimal postcontrol 1 of p & " " &
1839             ddecimal precontrol  2 of p & " " &
1840             ddecimal point      2 of p & " " &
1841             ddecimal postcontrol 2 of p & " " &
1842             ddecimal precontrol  3 of p & " " &
1843             ddecimal point      3 of p & " " &
1844             ddecimal postcontrol 3 of p & " " &
1845             ddecimal precontrol  0 of p
1846         fi
1847         withshadingcolors (a, b)
1848     )
1849     withshadingstep ( withshadingcolors (c, d) )
1850 enddef;
1851 def withtensorpatchinit (expr p, q, a, b, c, d) =

```

```

1852 withshadingstep(
1853     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1854     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1855     if string p: p & " " & q
1856     else:
1857         ddecimal point      0 of p & " " &
1858         ddecimal postcontrol 0 of p & " " &
1859         ddecimal precontrol  1 of p & " " &
1860         ddecimal point      1 of p & " " &
1861         ddecimal postcontrol 1 of p & " " &
1862         ddecimal precontrol  2 of p & " " &
1863         ddecimal point      2 of p & " " &
1864         ddecimal postcontrol 2 of p & " " &
1865         ddecimal precontrol  3 of p & " " &
1866         ddecimal point      3 of p & " " &
1867         ddecimal postcontrol 3 of p & " " &
1868         ddecimal precontrol  0 of p & " " &
1869         ddecimal point      0 of q & " " &
1870         ddecimal point      1 of q & " " &
1871         ddecimal point      2 of q & " " &
1872         ddecimal point      3 of q
1873     fi
1874     withshadingcolors (a, b)
1875 )
1876 withshadingstep ( withshadingcolors (c, d) )
1877 enddef;
1878 def withcoonspatchnext (expr f, p, a, b) =
1879     withshadingstep(
1880         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1881         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1882         if string p: p
1883         else:
1884             ddecimal postcontrol 1 of p & " " &
1885             ddecimal precontrol  2 of p & " " &
1886             ddecimal point      2 of p & " " &
1887             ddecimal postcontrol 2 of p & " " &
1888             ddecimal precontrol  3 of p & " " &
1889             ddecimal point      3 of p & " " &
1890             ddecimal postcontrol 3 of p & " " &
1891             ddecimal precontrol  0 of p
1892         fi
1893         withshadingcolors (a, b)
1894     )
1895 enddef;
1896 def withtensorpatchnext (expr f, p, q, a, b) =
1897     withshadingstep(
1898         withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1899         withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1900         if string p: p & " " & q

```

```

1901     else:
1902         ddecimal postcontrol 1 of p & " " &
1903         ddecimal precontrol 2 of p & " " &
1904         ddecimal point 2 of p & " " &
1905         ddecimal postcontrol 2 of p & " " &
1906         ddecimal precontrol 3 of p & " " &
1907         ddecimal point 3 of p & " " &
1908         ddecimal postcontrol 3 of p & " " &
1909         ddecimal precontrol 0 of p & " " &
1910         ddecimal point 0 of q & " " &
1911         ddecimal point 1 of q & " " &
1912         ddecimal point 2 of q & " " &
1913         ddecimal point 3 of q
1914     fi
1915     withshadingcolors (a, b)
1916 )
1917 enddef;
1918 def mplib_with_shade_method_analyze(expr p) =
1919     mplib_shade_path := p ;
1920     mplib_shade_step := 1 ;
1921     mplib_shade_fx := xpart point 0 of p ;
1922     mplib_shade_fy := ypart point 0 of p ;
1923     mplib_shade_lx := mplib_shade_fx ;
1924     mplib_shade_ly := mplib_shade_fy ;
1925     mplib_shade_nx := 0 ;
1926     mplib_shade_ny := 0 ;
1927     mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1928     mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1929     for i=1 upto length(p) :
1930         mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1931         mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1932         if mplib_shade_tx > mplib_shade_dx :
1933             mplib_shade_nx := i + 1 ;
1934             mplib_shade_lx := xpart point i of p ;
1935             mplib_shade_dx := mplib_shade_tx ;
1936         fi ;
1937         if mplib_shade_ty > mplib_shade_dy :
1938             mplib_shade_ny := i + 1 ;
1939             mplib_shade_ly := ypart point i of p ;
1940             mplib_shade_dy := mplib_shade_ty ;
1941         fi ;
1942     endfor ;
1943 enddef;
1944 vardef mplib_max_radius(expr p) =
1945     max (
1946         (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1947         (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1948         (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1949         (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)

```

```

1950 )
1951 enddef;
1952 def withshadingstep (text t) =
1953   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1954   withprescript "sh_step=" & decimal mplib_shade_step
1955   t
1956 enddef;
1957 let withfadestep = withshadingstep;
1958 def withshadingradius expr a =
1959   withprescript "sh_radius_a=" & decimal (xpart a)
1960   withprescript "sh_radius_b=" & decimal (ypart a)
1961 enddef;
1962 def withshadingorigin expr a =
1963   withprescript "sh_center_a=" & ddecimal a
1964   withprescript "sh_center_b=" & ddecimal a
1965 enddef;
1966 def withshadingvector expr a =
1967   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1968   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1969 enddef;
1970 def withshadingdirection expr a =
1971   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1972   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1973 enddef;
1974 def withshadingtransform expr a =
1975   withprescript "sh_transform=" & a
1976 enddef;
1977 def withshadingcenter expr a =
1978   withprescript "sh_center_a=" & ddecimal (
1979     center mplib_shade_path shifted (
1980       xpart a * xpart (llcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1981       ypart a * ypart (urcorner mplib_shade_path - llcorner mplib_shade_path)/2
1982     )
1983   )
1984 enddef;
1985 def withshadingcenters (expr a, b) =
1986   withprescript "sh_center_a=" & ddecimal a
1987   withprescript "sh_center_b=" & ddecimal b
1988   withshadingtransform "no"
1989   withshadingfactor 1
1990 enddef;
1991 let withshadingpoints = withshadingcenters;
1992 def withshadingextend (expr a, b) =
1993   withprescript "sh_extend=" &
1994     if a: "true" else: "false" fi & " " &
1995     if b: "true" else: "false" fi
1996 enddef;
1997 let withfadeextend = withshadingextend;
1998 def withshadingdomain expr d =

```

```

1999 withprescript "sh_domain=" & ddecimal d
2000 enddef;
2001 def withshadingfactor expr f =
2002   withprescript "sh_factor=" & decimal f
2003 enddef;
2004 def withshadingfraction expr a =
2005   if mplib_shade_step > 0 :
2006     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
2007   fi
2008 enddef;
2009 let withfadefraction = withshadingfraction;
2010 def withshadingcolors (expr a, b) =
2011   if mplib_shade_step > 0 :
2012     withprescript "sh_color=into"
2013     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
2014     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
2015   else :
2016     withprescript "sh_color=into"
2017     withprescript "sh_color_a=" & colordecimals a
2018     withprescript "sh_color_b=" & colordecimals b
2019   fi
2020 enddef;
2021 let withfadeopacity = withshadingcolors;
2022 def withshadingstroke expr a =
2023   withprescript "sh_stroking=" & a
2024 enddef;
2025 def withshadingmatrix expr s =
2026   withprescript "sh_matrix=" & s
2027 enddef;
2028 let withfadematrix = withshadingmatrix;
2029 def mpliblength primary t =
2030   runscript("return utf8.len[==[" & t & "]==]")
2031 enddef;
2032 def mplibsubstring expr p of t =
2033   runscript("return luamplib.unicodesubstring([==[" & t & "]==],"
2034     & decimal xpart p & ","
2035     & decimal ypart p & ")")
2036 enddef;
2037 def mlibuclength primary t =
2038   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
2039 enddef;
2040 def mlibucsubstring expr p of t =
2041   runscript("return luamplib.unicodesubstring([==[" & t & "]==],"
2042     & decimal xpart p & ","
2043     & decimal ypart p & ",true)")
2044 enddef;
2045 ]],
2046 legacyverbatimtex = [[
2047 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&"}") enddef;

```

```

2048 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
2049 let VerbatimTeX = specialVerbatimTeX;
2050 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
2051 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
2052 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
2053 "runscript(" &ditto&
2054 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
2055 "luamplib.in_the_fig=false" &ditto& ");";
2056 ]],
2057 texttextlabel = [[
2058 let luampliboriginalinfont = infont;
2059 primarydef s infont f =
2060   if (s < char 32)
2061     or (s = char 35) % #
2062     or (s = char 36) % $
2063     or (s = char 37) % %
2064     or (s = char 38) % &
2065     or (s = char 92) % \
2066     or (s = char 94) % ^
2067     or (s = char 95) % _
2068     or (s = char 123) % {
2069     or (s = char 125) % }
2070     or (s = char 126) % ~
2071     or (s = char 127) :
2072     s luampliboriginalinfont f
2073   else :
2074     rawtexttext(s)
2075   fi
2076 enddef;
2077 def fontsize expr f =
2078   begingroup
2079   save size; numeric size;
2080   size := mplibdimen("1em");
2081   if size = 0: 10pt else: size fi
2082 endgroup
2083 enddef;
2084 ]],
2085 }
2086

```

process_mplibcode

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

```

2087 luamplib.externalize = luamplib.externalize or { }
2088 local externalize = luamplib.externalize
2089

```

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

```

2090 luamplib.verbatiminput = false
2091 luamplib.everymplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
2092 luamplib.everyendmplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })

```

```

2093 function luamplib.process_mplibcode (data, instancename)
2094   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2095   if luamplib.legacyverbatim then
2096     luamplib.figid, tex_code_pre_mplib = 1, {}
2097   end
2098   local everymplib = luamplib.everymplib[instancename]
2099   local everyendmplib = luamplib.everyendmplib[instancename]
2100   data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
2101   :gsub("\r", "\n")

```

These lines are needed for mplibverbatim mode.

```

2102   if luamplib.verbatiminput then
2103     data = data:gsub("\mpcolor%s+{.-%b{}}", "mplibcolor(\"%1\")")
2104     :gsub("\mpdim%s+{%-b{}}", "mplibdimen(\"%1\")")
2105     :gsub("\mpdim%s+{\\%a+}", "mplibdimen(\"%1\")")
2106     data = scan_mpcode(data, replace_texblock)
2107   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.

```

2108     local t = { } -- to store btex, verbatimtex, string
2109     local function store (str)
2110       t[#t+1] = str
2111       return #t
2112     end
2113     data = scan_mpcode(data, {
2114       btex = function(str)
2115         return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", store(str)) -- space
2116       end,
2117       verbatimtex = function(str)
2118         return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", store(str)) -- semicolon
2119       end,
2120       str = function(str)
2121         return format("'\\unexpanded{!l!u!a!%s!m!p!l!}'", store(str))
2122       end,
2123       comment = function() return "" end,
2124       escapepercent = true,
2125     })
2126     run_tex_code(format("\mplibtmptoks\\expandafter{\\expanded{%s}}", data))
2127     data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

2128     :gsub("##", "#")
2129     :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2130   end
2131
2132   if externalize.running then

```

```

2133   if externalize.figure(data) then return end
2134 end
2135
2136 process(data, instancename)
2137 end
2138

```

pdfliterals 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.

```

2139 local figcontents = { post = { } }
2140 local function put2output(a,...)
2141   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2142 end
2143 local function pdf_startfigure(n,llx,lly,urx,ury)
2144   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2145 end
2146 local function pdf_stopfigure()
2147   put2output("\mplibstoptoPDF")
2148 end

```

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

```

2149 local function pdf_literalcode (...)
2150   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2151 end
2152 local start_pdf_code = pdfmode
2153 and function() pdf_literalcode"q" end
2154 or function() put2output"\special{pdf:bcontent}" end
2155 local stop_pdf_code = pdfmode
2156 and function() pdf_literalcode"Q" end
2157 or function() put2output"\special{pdf:econtent}" end
2158

```

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

```

2159 local function put_tex_boxes (object,prescript)
2160   local box = prescript.mplibtexboxid:explode":"
2161   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2162   if n and tw and th then
2163     local op = object.path
2164     local first, second, fourth = op[1], op[2], op[4]
2165     local tx, ty = first.x_coord, first.y_coord
2166     local sx, rx, ry, sy = 1, 0, 0, 1
2167     if tw ~= 0 then
2168       sx = (second.x_coord - tx)/tw
2169       rx = (second.y_coord - ty)/tw
2170       if sx == 0 then sx = 0.00001 end
2171     end
2172     if th ~= 0 then
2173       sy = (fourth.y_coord - ty)/th
2174       ry = (fourth.x_coord - tx)/th
2175       if sy == 0 then sy = 0.00001 end

```

```

2176     end
2177

```

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

```

2178     local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2179     put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2180 end
2181 end
2182

```

Colors

```

2183 local function do_preobj_CR(object,prescript)
2184   if object.postscript == "collect" then return end
2185   local override = prescript and prescript.mpliboverridecolor
2186   if override then
2187     pdf_literalcode(override)
2188     if not pdfmode and override:find" [cC][sS] " then
2189       local name1 = override:match("^/(.-) cs ")
2190       local name2 = override:match(" /(.-) CS ")
2191       if name1 then
2192         texpriint(ccexplat,
2193           "\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2194           name1, "}{\pdf_object_ref:n{", name1, "}}")
2195       )
2196     end
2197     if name2 and name1 ~= name2 then
2198       texpriint(ccexplat,
2199         "\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
2200         name2, "}{\pdf_object_ref:n{", name2, "}}")
2201     )
2202   end
2203 end
2204 else
2205   local cs = object.color
2206   if cs and #cs > 0 then
2207     pdf_literalcode(luamplib.colorconverter(cs))
2208   end
2209 end
2210 end
2211

```

For transparency, shading, fading, and pattern

```

2212 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2213 local pdfobjs, pdfetcs = {}, {}
2214 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2215 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2216 pdfetcs.pgfcspaces = "pgf@sys@addpdfresource@colorspaces@plain"
2217 local update_pdfobjs
2218 if pdfmode then
2219   function update_pdfobjs (os, stream)

```

```

2220     local key = os
2221     if stream then key = key..stream end
2222     local on = key and pdfobjs[key]
2223     if on then
2224         return on,false
2225     end
2226     if stream then
2227         on = pdf.immediateobj("stream",stream,os)
2228     elseif os then
2229         on = pdf.immediateobj(os)
2230     else
2231         on = pdf.reserveobj()
2232     end
2233     if key then
2234         pdfobjs[key] = on
2235     end
2236     return on,true
2237 end
2238 else
2239     function update_pdfobjs (os, stream, hex)
2240         local key = os
2241         if stream then key = key..stream end
2242         local on = key and pdfobjs[key]
2243         if on then
2244             return on,false
2245         end
2246         on = pdfetcs.cnt or 1
2247         if stream then
2248             if hex then
2249                 texsprintf(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2250             else
2251                 texsprintf(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2252             end
2253         elseif os then
2254             texsprintf(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2255         else
2256             texsprintf(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2257         end
2258         pdfetcs.cnt = on + 1
2259         if key then
2260             pdfobjs[key] = on
2261         end
2262         return on,true
2263     end
2264 end
2265 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2266 if pdfmode then
2267     pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2268     local getpagers = pdfetcs.getpagers

```

```

2269 local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2270 local initialize_resources = function (name)
2271   local tabname = format("%s_res",name)
2272   pdfetcs[tabname] = { }
2273   if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
2274     local obj = pdf.reserveobj()
2275     setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2276     luatexbase.add_to_callback("finish_pdffile", function()
2277       pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
2278     end,
2279     format("luamplib.%s.finish_pdffile",name))
2280   end
2281 end
2282 pdfetcs.fallback_update_resources = function (name, res)
2283   local tabname = format("%s_res",name)
2284   if not pdfetcs[tabname] then
2285     initialize_resources(name)
2286   end
2287   if luatexbase.callbacktypes.finish_pdffile then
2288     local t = pdfetcs[tabname]
2289     t[#t+1] = res
2290   else
2291     local tpr, n = getpagers() or "", 0
2292     tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2293     if n == 0 then
2294       tpr = format("%s/%s<<%s>>", tpr, name, res)
2295     end
2296     setpagers(tpr)
2297   end
2298 end
2299 else
2300   texsprint (
2301     "\\luamplibatfirstshipout{",
2302     "\\special{pdf:obj @MPlibTr<<>>}",
2303     "\\special{pdf:obj @MPlibSh<<>>}",
2304     "\\special{pdf:obj @MPlibCS<<>>}",
2305     "\\special{pdf:obj @MPlibPt<<>>}}")
2306 )
2307 pdfetcs.fallback_update_resources = function (name,res,obj)
2308   texsprint("\\special{pdf:put ", obj, " <<", res, ">>}")
2309   local tabname = format("%s_res",name)
2310   if not pdfetcs[tabname] then
2311     texsprint("\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}")
2312     pdfetcs[tabname] = { }
2313   end
2314   tableinsert(pdfetcs[tabname], res)
2315 end
2316 end
2317

```

Transparency

```

2318 local function add_extgs_resources (on, new)
2319   local key = format("MPLibTr%s", on)
2320   if new then
2321     local val = format(pdfetcs.resfmt, on)
2322     if pdfmanagement then
2323       texsprintf (
2324         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2325       )
2326     else
2327       local tr = format("/%s %s", key, val)
2328       if is_defined(pdfetcs.pgfbextgs) then
2329         texsprintf ( "\\csname ", pdfetcs.pgfbextgs, "\\endcsname{", tr, "}" )
2330       elseif is_defined"TRP@list" then
2331         texsprintf(catat11,
2332           [[\if@files\immediate\write\@auxout{]],
2333           [[\string\g@addto@macro\string\TRP@list{]],
2334           tr,
2335           [[}]\fi]]
2336         )
2337         if not get_macro"TRP@list":find(tr) then
2338           texsprintf(catat11,[[\global\TRP@reruntrue]])
2339         end
2340       else
2341         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2342       end
2343     end
2344   end
2345   return key
2346 end
2347
2348 local do_preobj_TR
2349 do
2350   local transparency_modes = {
2351     [0] = "Normal",
2352     "Normal",      "Multiply",    "Screen",      "Overlay",
2353     "SoftLight",   "HardLight",   "ColorDodge",  "ColorBurn",
2354     "Darken",      "Lighten",    "Difference",  "Exclusion",
2355     "Hue",         "Saturation", "Color",       "Luminosity",
2356     "Compatible",
2357     normal        = "Normal",    multiply       = "Multiply",   screen        = "Screen",
2358     overlay       = "Overlay",    softlight     = "SoftLight",  hardlight     = "HardLight",
2359     colordodge    = "ColorDodge", colorburn     = "ColorBurn",  darken        = "Darken",
2360     lighten       = "Lighten",    difference    = "Difference", exclusion     = "Exclusion",
2361     hue           = "Hue",        saturation    = "Saturation", color         = "Color",
2362     luminosity    = "Luminosity", compatible     = "Compatible",
2363   }
2364   local full_opacity
2365   local function get_full_opacity()

```

```

2366     local on, new = update_pdfobjs"<</BM/Normal/ca 1/CA 1/AIS false>>"
2367     local key = add_extgs_resources(on,new)
2368     full_opacity = format("/%s gs",key)
2369     return full_opacity
2370 end
2371 function do_preobj_TR(object,prescript)
2372     if object.postscript == "collect" then return end
2373     local opaq = prescript and prescript.tr_transparency
2374     if not opaq then return end
2375
2376     local key, on, os, new
2377     local mode = prescript.tr_alternative or 1
2378     mode = transparency_modes[tonumber(mode) or mode:lower()]
2379     if not mode then
2380         mode = prescript.tr_alternative
2381         warn("unsupported blend mode: '%s'", mode)
2382     end
2383     opaq = opaq:explode"."
2384     for i,v in ipairs(opaq) do
2385         opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2386     end
2387     os = format("<</BM/%s/ca %s/CA %s/AIS false>>",mode,opaq[1],opaq[2] or opaq[1])
2388     on, new = update_pdfobjs(os)
2389     key = add_extgs_resources(on,new)
2390     pdf_literalcode("/%s gs",key)
2391     return full_opacity or get_full_opacity()
2392 end
2393 end
2394

```

Shading with *metafun* format.

```

2395 local function add_shading_resources (on, new)
2396     if new then
2397         local key, val = format("MPLibSh%s", on), format(pdfetcs.resfmt, on)
2398         if pdfmanagement then
2399             texsprint (
2400                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2401             )
2402         else
2403             local res = format("/%s %s", key, val)
2404             pdfetcs.fallback_update_resources("Shading",res,"@MPLibSh")
2405         end
2406     end
2407 end
2408 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2409     for _,v in ipairs{ca,cb} do
2410         for i,vv in ipairs(v) do
2411             for ii,vvv in ipairs(vv) do
2412                 v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv

```

```

2413     end
2414 end
2415 end
2416 local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2417 if steps > 1 then
2418     local list,bounds,encode = { },{ },{ }
2419     for i=1,steps do
2420         if i < steps then
2421             bounds[i] = format("%.3f", fractions[i] or 1)
2422         end
2423         encode[2*i-1] = 0
2424         encode[2*i] = 1
2425         os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
2426         :gsub(decimals,rmzeros)
2427         list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2428     end
2429     os = tableconcat {
2430         "<</FunctionType 3",
2431         format("/Bounds[%s]", tableconcat(bounds, ' ')),
2432         format("/Encode[%s]", tableconcat(encode, ' ')),
2433         format("/Functions[%s]", tableconcat(list, ' ')),
2434         format("/Domain[%s]>>", domain),
2435     } :gsub(decimals,rmzeros)
2436 else
2437     os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
2438     :gsub(decimals,rmzeros)
2439 end
2440 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2441 os = tableconcat {
2442     format("<</ShadingType %i", shtype),
2443     format("/ColorSpace %s", colorspace),
2444     format("/Function %s", objref),
2445     format("/Coords[%s]", coordinates),
2446     format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2447 } :gsub(decimals,rmzeros)
2448 local on, new = update_pdfobjs(os)
2449 add_shading_resources(on, new)
2450 return on
2451 end
2452
2453 local function get_mp_matrix (matrix)
2454     process(("mplibtransformmatrix(%s);"):format(matrix), "@mplibtransformmatrix")
2455     return luamplib.transformmatrix
2456 end
2457
2458 local do_preobj_SH
2459 do
2460     pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2461         run_tex_code({

```

```

2462     [[\color_model_new:nnn]],
2463     format("{mplibcolorspace_%s}", names:gsub(",","_")),
2464     format("{DeviceN}{names={%s}}", names),
2465     [[\mplibmptoks\expanded{{\pdf_object_ref_last:}}]],
2466   }, ccexplat)
2467   local colorspace = texgettoks"mplibmptoks"
2468   t[names] = colorspace
2469   return colorspace
2470 end })
2471 local function color_normalize(ca,cb)
2472   if #cb == 1 then
2473     if #ca == 4 then
2474       cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2475     else -- #ca = 3
2476       cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2477     end
2478   elseif #cb == 3 then -- #ca == 4 : rgb to cmyk
2479     local c, m, y = 1-cb[1], 1-cb[2], 1-cb[3]
2480     local k = math.min(c, m, y)
2481     if k == 1 then
2482       c, m, y = 0, 0, 0
2483     else
2484       c, m, y = (c-k)/(1-k), (m-k)/(1-k), (y-k)/(1-k)
2485     end
2486     cb[1], cb[2], cb[3], cb[4] = c, m, y, k
2487   end
2488 end
2489 local function write_mesh_objs (shtype, colorspace, stream, devicen, perrow)
2490   local cc = colorspace == "/DeviceCMYK" and 4
2491     or colorspace == "/DeviceRGB" and 3
2492     or devicen or 1
2493   local dict = format(
2494     "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2495     shtype,
2496     perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2497     colorspace,
2498     (" 0 1"):rep(cc))
2499   local on, new
2500   if pdfmode then
2501     on, new = update_pdfobjs(dict, stream)
2502   else
2503     stream = format("%02X"):rep(stream:len()), stream:byte(1,stream:len())
2504     on, new = update_pdfobjs(dict, stream, true) -- hex is true
2505   end
2506   add_shading_resources(on, new)
2507   return on
2508 end
2509 local function colors_ff (colors)
2510   local t, devicen = { }

```

```

2511     for i,v in ipairs(colors) do
2512         t[i] = { }
2513         for _,vv in ipairs(v) do
2514             local tt = tableconcat(vv," "):explode()
2515             for _,vuv in ipairs(tt) do
2516                 tableinsert(t[i], string.char(math.floor(vuv * 0xFF)))
2517             end
2518             devicen = #tt
2519         end
2520     end
2521     return t, devicen
2522 end
2523 local function coords_ffff (coords)
2524     local Xt, Yt = { }, { }
2525     for i,v in ipairs(coords) do
2526         for ii,vv in ipairs(v) do
2527             local t = ii % 2 == 1 and Xt or Yt
2528             t[#t+1] = tonumber(vv)
2529         end
2530     end
2531     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2532     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2533     local wd, ht = xmax - xmin, ymax - ymin
2534     for i,v in ipairs(coords) do
2535         for ii,vv in ipairs(v) do
2536             local min = ii % 2 == 1 and xmin or ymin
2537             local dim = ii % 2 == 1 and wd or ht
2538             coords[i][ii] = string.pack(">H", math.floor((vv - min)/dim * 0xFFFF ))
2539         end
2540     end
2541     return coords, xmin, ymin, wd, ht
2542 end
2543 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2544     local perrow = prescript.sh_lattice_perrow or 2
2545     local steps = tonumber(prescript.sh_step) or 0
2546     local coords, colors = { }, { }
2547     if steps < 4 then
2548         local path = object.path
2549         for _,i in ipairs{1,2,4,3} do
2550             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2551         end
2552         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}}, {{1,1,0}} }
2553         colorspace = "/DeviceRGB"
2554     else
2555         local t = prescript.sh_lattice_data:explode()
2556         for i = 1, #t, 2 do
2557             coords[#coords+1] = { t[i], t[i+1] }
2558         end
2559         for i = 1, steps do

```

```

2560         if not ca[i] then err"colorspace mismatch?" end
2561         colors[#colors+1] = { ca[i] }
2562     end
2563 end
2564 if #coords % perrow ~= 0 then err"vertices_per_row mismatches lattice_coords" end
2565
2566 local stream, xmin, ymin, wd, ht, devicen = { }
2567 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2568 colors, devicen = colors_ff(colors)
2569 for i = 1, #coords do
2570     stream[#stream+1] = tableconcat(coords[i])
2571     stream[#stream+1] = tableconcat(colors[i])
2572 end
2573
2574 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2575 local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2576 return on, matrix
2577 end
2578 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2579     local steps = tonumber(prescript.sh_step) or 0
2580     local coords, colors, edges = { }, { }, { }
2581     if steps < 3 then
2582         local path = object.path
2583         for i = 1, 3 do
2584             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2585         end
2586         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}} }
2587         edges = { 0, 0, 0 }
2588         colorspace = "/DeviceRGB"
2589     else
2590         for i = 1, steps do
2591             local t = prescript["sh_triangle_vertex_" .. i]:explode()
2592             if #t == 2 then
2593                 coords[#coords+1] = { t[1], t[2] }
2594             elseif #t == 6 then
2595                 coords[#coords+1] = { t[1], t[2] }
2596                 coords[#coords+1] = { t[3], t[4] }
2597                 coords[#coords+1] = { t[5], t[6] }
2598             end
2599             if not ca[i] then err"colorspace mismatch?" end
2600             colors[#colors+1] = { ca[i] }
2601             edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])
2602         end
2603     end
2604
2605     local stream, xmin, ymin, wd, ht, devicen = { }
2606     coords, xmin, ymin, wd, ht = coords_ffff(coords)
2607     colors, devicen = colors_ff(colors)
2608     for i = 1, #coords do

```

```

2609     stream[#stream+1] = string.char(edges[i])
2610     stream[#stream+1] = tableconcat(coords[i])
2611     stream[#stream+1] = tableconcat(colors[i])
2612 end
2613
2614 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2615 local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2616 return on, matrix
2617 end
2618 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2619     local tensor = prescript.sh_type == "tensor"
2620     local steps = tonumber(prescript.sh_step) or 0
2621     local coords, colors, edges = { }, { }, { }
2622     if steps < 2 then
2623         local path, t = object.path, { }
2624         for i = 1, 4 do
2625             t[#t+1] = path[i].x_coord
2626             t[#t+1] = path[i].y_coord
2627             t[#t+1] = path[i].right_x
2628             t[#t+1] = path[i].right_y
2629             local j = i == 4 and 1 or i+1
2630             t[#t+1] = path[j].left_x
2631             t[#t+1] = path[j].left_y
2632         end
2633         if tensor then
2634             t = table.move(prescript.sh_tensor_path:explode(), 1, 8, #t+1, t)
2635         end
2636         coords = { t }
2637         colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2638         edges = { 0 }
2639         colorspace = "/DeviceRGB"
2640     else
2641         for i = 1, steps do
2642             local edge = tonumber(prescript["sh_coons_edge_"..i])
2643             if edge then
2644                 edges[#edges+1] = edge
2645                 coords[#coords+1] = prescript["sh_coons_path_"..i]:explode()
2646                 if edge == 0 then
2647                     if not (ca[i] and cb[i] and ca[i+1] and cb[i+1]) then err"colorspace mismatch?" end
2648                     colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2649                 else
2650                     if not (ca[i] and cb[i]) then err"colorspace mismatch?" end
2651                     colors[#colors+1] = { ca[i], cb[i] }
2652                 end
2653             end
2654         end
2655     end
2656     local stream, xmin, ymin, wd, ht, devicen = { }

```

```

2658 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2659 colors, devicen = colors_ff(colors)
2660 for i = 1, #coords do
2661     stream[#stream+1] = string.char(edges[i])
2662     stream[#stream+1] = tableconcat(coords[i])
2663     stream[#stream+1] = tableconcat(colors[i])
2664 end
2665
2666 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2667 local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2668 return on, matrix
2669 end
2670 function do_preobj_SH(object, prescript)
2671     local shade_no
2672     local sh_type = prescript and prescript.sh_type
2673     if not sh_type then return end
2674
2675     local domain = prescript.sh_domain or "0 1"
2676     local centera = (prescript.sh_center_a or "0 0"):explode()
2677     local centerb = (prescript.sh_center_b or "0 0"):explode()
2678     local transform = prescript.sh_transform == "yes"
2679     local sx,sy,sr,dx,dy = 1,1,1,0,0
2680     if transform then
2681         local first = (prescript.sh_first or "0 0"):explode()
2682         local setx = (prescript.sh_set_x or "0 0"):explode()
2683         local sety = (prescript.sh_set_y or "0 0"):explode()
2684         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2685         if x ~= 0 and y ~= 0 then
2686             local path = object.path
2687             local path1x = path[1].x_coord
2688             local path1y = path[1].y_coord
2689             local path2x = path[x].x_coord
2690             local path2y = path[y].y_coord
2691             local dxa = path2x - path1x
2692             local dya = path2y - path1y
2693             local dxb = setx[2] - first[1]
2694             local dyb = sety[2] - first[2]
2695             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2696                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
2697                 sy = dya / dyb ; if sy < 0 then sy = - sy end
2698                 sr = math.sqrt(sx^2 + sy^2)
2699                 dx = path1x - sx*first[1]
2700                 dy = path1y - sy*first[2]
2701             end
2702         end
2703     end
2704     local ca, cb, colorspace, steps, fractions
2705     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:"" }
2706     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:"" }

```

```

2707 steps = tonumber(prescript.sh_step) or 1
2708 if steps > 1 then
2709     fractions = { prescript.sh_fraction_1 or 0 }
2710     for i=2,steps do
2711         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2712         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode":"
2713         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode":"
2714     end
2715 end
2716 if prescript.mplib_spotcolor then
2717     ca, cb = { }, { }
2718     local names, pos, objref = { }, -1, ""
2719     local script = object.prescript:explode"\13+"
2720     for i=#script,1,-1 do
2721         local name, value
2722         if script[i]:find"mplib_spotcolor" then
2723             local t = script[i]:explode"="[2]:explode":"
2724             value, objref, name = t[1], t[2], t[3]
2725         elseif script[i]:find"mplib_texcolor" then
2726             local t = script[i]:explode"="[2]:explode"!"
2727             name, value = t[1], (t[2] or 100)/100
2728         end
2729         if name and value then
2730             if not names[name] then
2731                 pos = pos+1
2732                 names[name] = pos
2733                 names[#names+1] = name
2734             end
2735             local t = { }
2736             for j=1,names[name] do t[#t+1] = 0 end
2737             t[#t+1] = value
2738             tableinsert(#ca == #cb and ca or cb, t)
2739         end
2740     end
2741     for _,t in ipairs{ca,cb} do
2742         for _,tt in ipairs(t) do
2743             for i=1,#names-#tt do tt[#tt+1] = 0 end
2744         end
2745     end
2746     if #names == 1 then
2747         colorspace = objref
2748     else
2749         colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2750     end
2751 else
2752     local model = 0
2753     for _,t in ipairs{ca,cb} do
2754         for _,tt in ipairs(t) do
2755             model = model > #tt and model or #tt

```

```

2756     end
2757 end
2758 for _,t in ipairs{ca,cb} do
2759     for _,tt in ipairs(t) do
2760         if #tt < model then
2761             color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2762         end
2763     end
2764 end
2765 colorspace = model == 4 and "/DeviceCMYK"
2766             or model == 3 and "/DeviceRGB"
2767             or model == 1 and "/DeviceGray"
2768             or err"unknown color model"
2769 end
2770 local extend = prescript.sh_extend
2771 local mesh_matrix
2772 if sh_type == "linear" then
2773     local coordinates = format("%f %f %f %f",
2774         dx + sx*centera[1], dy + sy*centera[2],
2775         dx + sx*centerb[1], dy + sy*centerb[2])
2776     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2777 elseif sh_type == "circular" then
2778     local factor = prescript.sh_factor or 1
2779     local radiusa = factor * prescript.sh_radius_a
2780     local radiusb = factor * prescript.sh_radius_b
2781     local coordinates = format("%f %f %f %f %f %f",
2782         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2783         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2784     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2785 elseif sh_type == "coons" or sh_type == "tensor" then
2786     shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2787 elseif sh_type == "triangle" then
2788     shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2789 elseif sh_type == "lattice" then
2790     shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2791 else
2792     err"unknown shading type"
2793 end
2794
2795 local matrix = prescript.sh_matrix
2796 if matrix and matrix:find"%a" then
2797     local t = get_mp_matrix(matrix)
2798     matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2799 end
2800 if mesh_matrix and matrix then
2801     local a, b = mesh_matrix:explode(), matrix:explode()
2802     matrix = format("%f %f %f %f %f %f",
2803         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],
2804         a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)

```

```

2805     end
2806
2807     return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix
2808 end
2809 end
2810

```

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

```

2811 local function add_pattern_resources (key, val)
2812   if pdfmanagement then
2813     texsprint (
2814       "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2815     )
2816   else
2817     local res = format("/%s %s", key, val)
2818     if is_defined(pdfetcs.pgfpattern) then
2819       texsprint ( "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" )
2820     else
2821       pdfetcs.fallback_update_resources("Pattern",res,"@MPlibPt")
2822     end
2823   end
2824 end
2825 if pdfmode then
2826   function luamplib.dolatelua (on, os, matrix)
2827     local h, v = pdf.getpos()
2828     local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2829     matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2830     :gsub(decimals,rmzeros)
2831     pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2832     pdf.refobj(on)
2833   end
2834 else
2835   pdfetcs.shadingpatterns = { }
2836   pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2837   function luamplib.dolatelua (on, kind, xobj)
2838     local h, v
2839     local t = pdfetcs.shadingpatterns[on] or { }
2840     local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2841       or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2842     if shift then
2843       h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2844     else
2845       h, v = pdf.getpos()
2846       h = format("%f", h/factor) :gsub(decimals,rmzeros)
2847       v = format("%f", v/factor) :gsub(decimals,rmzeros)
2848     end
2849     if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2850       warn"Rerun to get correct shading pattern"
2851     end

```

```

2852 local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2853 local init = pdfetcs.shadingpatterninit_w
2854 if init then pdfetcs.shadingpatterninit_w = nil end
2855 local f = ioopen(name, init and "w" or "a")
2856 if f then
2857     f:write(("s %s %s\n"):format(on, h, v))
2858     f:close()
2859 else
2860     err"cannot write a file. check the cache dir path"
2861 end
2862 end
2863 end
2864 local function do_preobj_shading (object, prescript)
2865 if not prescript or not prescript.sh_operand_type then return end
2866 local on,_,matrix = do_preobj_SH(object, prescript)
2867 local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2868 matrix = matrix or "1 0 0 1 0 0"
2869 if not pdfmode and prescript.sh_in_xobj == "yes" then -- only in dvi mode
2870     on = update_pdfobjs(("<<Matrix[%s]>>"):format(os, matrix))
2871     goto skip_latelua
2872 end
2873 on = update_pdfobjs()
2874 if pdfmode then
2875     put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"],[",matrix,"]]}")})
2876 else
2877     local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2878               or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2879     local init = pdfetcs.shadingpatterninit_r
2880     if init then
2881         pdfetcs.shadingpatterninit_r = nil
2882         local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2883         local f = ioopen(name)
2884         if f then
2885             for line in f:lines() do
2886                 local t = line:explode()
2887                 pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2888             end
2889             f:close()
2890         end
2891     end

```

This seems to be needed for proper functioning:

```

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

2892 local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2893 local mt = matrix:explode()
2894 matrix = format("s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2895 texsprint( "\\special{pdf:put ", format(pdfetcs.resfmt, on),

```

```

2896         format(" <<%s/Matrix[%s]>>}", os, matrix) )
2897     put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2898         xobj and ("'%s',[[%s]]"):format(xobj[1], xobj[2]))
2899 end
2900 ::skip_latelua::
2901 local key, val = format("MPLibPt%s", on), format(pdfetcs.resfmt, on)
2902 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.

```

2903 local stroke = prescript.sh_stroking
2904 if stroke == "filldraw" or stroke == "both" then
2905     pdf_literalcode("/Pattern cs/%s scn /Pattern CS/%s SCN", key, key)
2906 elseif stroke == "draw" or stroke == "yes" then
2907     pdf_literalcode("/Pattern CS/%s SCN", key)
2908 else
2909     pdf_literalcode("/Pattern cs/%s scn", key)
2910 end

```

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

```

2911 prescript.sh_type = nil
2912 end
2913

```

Tiling Patterns

```

2914 pdfetcs.patterns = { }
2915 local function gather_resources (optres, ispattern)
2916     local t = { }
2917     if pdfmanagement then
2918         for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2919             local mytoks
2920             run_tex_code ({
2921                 "\\mplibtmptoks\\expanded{",
2922                 "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/", v, "}",
2923                 "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/", v, "}}", "}",
2924             }, ccexplat)
2925             mytoks = texgettoks"mplibtmptoks"
2926             if not pdfmode then
2927                 mytoks = mytoks:gsub("\\str_convert_pdfname:n%s*{(.-)}", "%1") -- why not expanded?
2928             end
2929             mytoks = mytoks and mytoks:gsub("^%s*(.)%s*$", "%1")
2930             if mytoks and mytoks ~= "" then
2931                 t[#t+1] = ("%s<<%s>>"):format(v, mytoks)
2932             end
2933         end
2934     elseif is_defined(pdfetcs.pgftextgs) then
2935         run_tex_code "\\relax" -- flush tex.sprint queue
2936         if pdfmode then
2937             for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2938                             ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",

```

```

2939         Pattern      = "pgf@sys@pgf@resource@list@patterns", } do
2940     local res = (get_macro(v) or ""):gsub("^%s*(.)%s*$", "%1")
2941     if res ~= "" then
2942         t[#t+1] = ("%s<<%s>>"):format(k, res )
2943     end
2944 end
2945 else
2946     local abc = get_macro"pgfutil@abc" or ""
2947     for k,v in pairs { ExtGState = "@pgfextgs",
2948         ColorSpace = "@pgfcolorspaces",
2949         Pattern      = "@pgfpatterns", } do
2950         local tt = { }
2951         for vv in abc:gmatch( v .. "%s*(%b<>)" ) do
2952             tt[#tt+1] = vv:match("^<<%s*(.)%s*>>$")
2953         end
2954         if #tt > 0 then
2955             t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2956         end
2957     end
2958 end

```

We still have to deal with Shading resources.

```

2959     if luatexbase.callbacktypes.finish_pdffile then
2960         if pdfetcs.Shading_res then
2961             t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2962         end
2963     else
2964         local res = pdfetcs.getpageres()
2965         res = res and res:match"/Shading%s*%b<>"
2966         if res then
2967             t[#t+1] = res
2968         end
2969     end
2970 else
2971     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

```

2972     warn"transparent package is not fully functional without pdfmanagement code."
2973 end
2974 if luatexbase.callbacktypes.finish_pdffile then
2975     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2976         local tt = pdfetcs[v.."_res"]
2977         if tt then
2978             t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2979         end
2980     end
2981 else
2982     local res = pdfetcs.getpageres()
2983     if res then

```

```

2984         t[#t+1] = res
2985     end
2986 end
2987 end
2988 local result = tableconcat(t)
2989 if optres ~= "" then
2990     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2991         local res = optres:match("/"..v.."s*%b<>")
2992         if res then
2993             if result:find("/"..v) then
2994                 res = res:match("<<(.+)>>$")
2995                 result = result:gsub("/"..v.."s*<<", "%1"..res, 1)
2996             else
2997                 result = result .. res
2998             end
2999         end
3000     end
3001 end
3002 return result
3003 end
3004 function luamplib.registerpattern ( boxid, name, opts )
3005     local box = texgetbox(boxid)
3006     local wd = format("%.3f", box.width/factor)
3007     local hd = format("%.3f", (box.height+box.depth)/factor)
3008     info("w/h/d of pattern '%s': %s 0", name, format("%s %s", wd, hd):gsub(decimals,rmzeros))
3009     if opts.xstep == 0 then opts.xstep = nil end
3010     if opts.ystep == 0 then opts.ystep = nil end
3011     if opts.colored == nil then
3012         opts.colored = opts.coloured
3013         if opts.colored == nil then
3014             opts.colored = true
3015         end
3016     end
3017     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
3018     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
3019     if opts.matrix and opts.matrix:find"%a" then
3020         local t = get_mp_matrix(opts.matrix)
3021         opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
3022         opts.xshift = opts.xshift or format("%f", t[5])
3023         opts.yshift = opts.yshift or format("%f", t[6])
3024     end
3025     local attr = {
3026         "/Type/Pattern",
3027         "/PatternType 1",
3028         format("/PaintType %i", opts.colored and 1 or 2),
3029         "/TilingType 2",
3030         format("/XStep %s", opts.xstep or wd),
3031         format("/YStep %s", opts.ystep or hd),
3032         format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),

```

```

3033 }
3034 local optres = opts.resources or ""
3035 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
3036 local patterns = pdfetcs.patterns
3037 if pdfmode then
3038   if opts.bbox then
3039     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3040   end
3041   attr = tableconcat(attr) :gsub(decimals,rmzeros)
3042   local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
3043   patterns[name] = { id = index, colored = opts.colored }
3044 else
3045   local cnt = #patterns + 1
3046   local objname = "@mplibpattern" .. cnt
3047   local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
3048   texsprint (
3049     "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
3050     "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
3051     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3052     "\\special{pdf:bcontent}",
3053     "\\special{pdf:bxobj ", objname, " ", metric, "}",
3054     "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
3055     "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
3056     "\\special{pdf:put @resources <<", optres, ">>}",
3057     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3058     "\\special{pdf:econtent}}")
3059   )
3060   patterns[cnt] = objname
3061   patterns[name] = { id = cnt, colored = opts.colored }
3062   patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
3063 end
3064 end
3065
3066 local do_preobj_PAT
3067 do
3068   local function pattern_colorspace (cs)
3069     local on, new = update_pdfobjs(format("[Pattern %s]", cs))
3070     if new then
3071       local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
3072       if pdfmanagement then
3073         texsprint (
3074           "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
3075         )
3076       else
3077         local res = format("/%s %s", key, val)
3078         if is_defined(pdfetcs.pgfcolorspace) then
3079           texsprint ( "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" )
3080         else
3081           pdfetcs.fallback_update_resources("ColorSpace",res,"@MPlibCS")
3082         end
3083       end
3084     end
3085   end
3086 end

```

```

3082     end
3083   end
3084 end
3085 return on
3086 end
3087 local function uncoloredGS(color,key,fill)
3088   local cs
3089   if color:find" cs " then
3090     local name
3091     if fill then
3092       name, color = color:match"^/(.-) cs (.-) sc"
3093     else
3094       name, color = color:match" /(-) CS (-) SC"
3095     end
3096     if pdfmode then
3097       cs = format("%s 0 R", ltx.pdf.object_id( name ))
3098       if cs == "0 0 R" then -- assumes colorspace.sty
3099         _, cs = get_spc_name_obj("/"..name)
3100       end
3101     else
3102       run_tex_code({ "\\mplibmptoks\\expanded{{\\pdf_object_ref:n{", name, "}}}" }, ccexplat)
3103       cs = texgettoks"mplibmptoks"
3104     end
3105   else
3106     if fill then
3107       color = colorsplit(color)
3108     else
3109       color = color:match"%a+ (.-) %a+":explode()
3110     end
3111     cs = #color == 4 and "/DeviceCMYK" or #color == 3 and "/DeviceRGB" or "/DeviceGray"
3112     color = tableconcat(color," ")
3113   end
3114   return fill and format("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3115     or format("/MPLibCS%i CS %s /%s SCN", pattern_colorspace(cs), color, key)
3116 end
3117 function do_preobj_PAT(object, prescript)
3118   local name = prescript and prescript.mplibpattern
3119   if not name then return end
3120   local patterns = pdfetcs.patterns
3121   local patt = patterns[name]
3122   local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3123   local key = format("MPLibPt%s",index)
3124   local stroke, fillgs, strkgs = prescript.mplibpatternstroke
3125   if patt.colored then
3126     fillgs = format("/Pattern cs /%s scn", key)
3127     strkgs = stroke and format("/Pattern CS /%s SCN", key)
3128   else
3129     local color = prescript.mpliboverridecolor
3130     if not color then

```

```

3131     local t = object.color
3132     color = t and #t>0 and luamplib.colorconverter(t)
3133     end
3134     if not color then return end
3135     fillgs = uncoloredGS(color,key,true)
3136     strkgs = stroke and uncoloredGS(color,key,false)
3137     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.

```

3138     if stroke == "filldraw" or stroke == "both" then
3139         pdf_literalcode("%s %s", fillgs, strkgs)
3140     elseif stroke == "draw" or stroke == "yes" then
3141         pdf_literalcode(strkgs)
3142     else
3143         pdf_literalcode(fillgs)
3144     end
3145     if not patt.done then
3146         local val = pdfmode and format("%s 0 R",index) or patterns[index]
3147         add_pattern_resources(key,val)
3148     end
3149     patt.done = true
3150 end
3151 end
3152

```

Fading

```

3153 pdfetcs.fading = { }
3154 local function do_preobj_FADE (object, prescript)
3155     local fd_type = prescript and prescript.mplibfadetype
3156     local fd_stop = prescript and prescript.mplibfadestate
3157     if not fd_type then
3158         return fd_stop -- returns "stop" (if picture) or nil
3159     end
3160     local on, os, new
3161     if fd_type == "masking" then
3162         local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3163         on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
3164         local bc = prescript.mplibmaskingbgcolor
3165         bc = bc and bc:gsub(":", " ")
3166         bc = bc and ("/BC[%s]"):format(bc):gsub(decimals,rmzeros) or ""
3167         os = format("<</SMask<</S/Luminosity/G %s%>>>>",
3168             pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3169     else
3170         local bbox = prescript.mplibfadebbox:explode":"
3171         local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
3172         if not vec then
3173             if fd_type == "linear" then
3174                 vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3175             else

```

```

3176     local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3177     vec = {centerx, centery, centerx, centery} -- center for both circles
3178     end
3179 end
3180 local coords = { vec[1], vec[2], vec[3], vec[4] }
3181 if fd_type == "linear" then
3182     coords = format("%f %f %f %f", tableunpack(coords))
3183 elseif fd_type == "circular" then
3184     local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3185     local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":""
3186     tableinsert(coords, 3, radius[1])
3187     tableinsert(coords, radius[2])
3188     coords = format("%f %f %f %f %f %f", tableunpack(coords))
3189 else
3190     err("unknown fading method '%s'", fd_type)
3191 end
3192 fd_type = fd_type == "linear" and 2 or 3
3193 local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3194 local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":"" }
3195 local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":"" }
3196 if steps > 1 then
3197     fractions = { prescript.sh_fraction_1 or 0 }
3198     for i=2,steps do
3199         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3200         ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":""
3201         cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":""
3202     end
3203 end
3204
3205 local wd = prescript.mplibadjustbbox
3206 if not wd then
3207     local pen = mplib.pen_info(object)
3208     if pen and pen.width then
3209         wd = pen.width / 2 * math.sqrt(2)
3210     end
3211 end
3212 if wd then
3213     bbox = { bbox[1]-wd, bbox[2]-wd, bbox[3]+wd, bbox[4]+wd }
3214 end
3215
3216 local dx, dy = -bbox[1], -bbox[2]
3217 local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3218 matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3219 matrix[5] = matrix[5] + dx
3220 matrix[6] = matrix[6] + dy
3221 matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3222 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3223 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3224 on = update_pdfobjs(os)

```

```

3225     bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3226     local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
3227         :gsub(decimals,rmzeros)
3228     os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3229     on = update_pdfobjs(os)
3230     local resources = format(pdfetcs.resfmt, on)
3231     on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
3232     local attr = tableconcat{
3233         "/Subtype/Form",
3234         "/BBox[" .. bbox .. "]",
3235         "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
3236         "/Resources " .. resources,
3237         "/Group " .. format(pdfetcs.resfmt, on),
3238     } :gsub(decimals,rmzeros)
3239     on = update_pdfobjs(attr, streamtext)
3240     os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
3241 end
3242 on, new = update_pdfobjs(os)
3243 local key = add_extgs_resources(on,new)
3244 start_pdf_code()
3245 pdf_literalcode("/%s gs", key)
3246 if fd_stop then return "standalone" end
3247 return "start"
3248 end
3249

```

Transparency Group

```

3250 pdfetcs.tr_group = { shifts = { } }
3251 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3252 local function do_preobj_GRP (object, prescript)
3253     local grstate = prescript and prescript.gr_state
3254     if not grstate then return end
3255     local trgroup = pdfetcs.tr_group
3256     if grstate == "start" then
3257         trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3258         trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3259         trgroup.wrapped = false
3260         for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+)") do
3261             trgroup[v] = true
3262         end
3263         trgroup.bbox = prescript.mplibgroupbbox:explode": "
3264         trgroup.adjustbbox = prescript.mplibadjustbbox
3265         if not trgroup.adjustbbox then
3266             trgroup.widths = { }
3267         end
3268         put2output[["\beginpgroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3269     elseif grstate == "stop" then
3270         local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3271

```

```

3272 local wd = trgroup.adjustbbox
3273 if not wd then
3274   if #trgroup.widths > 0 then
3275     wd = math.max(tableunpack(trgroup.widths))
3276     if wd > 0 then
3277       wd = wd / 2 * math.sqrt(2)
3278     end
3279   end
3280   trgroup.widths = nil
3281 end
3282 if wd then
3283   llx,lly,urx,ury = llx-wd, lly-wd, urx+wd, ury+wd
3284 end
3285
3286 put2output(tableconcat{
3287   "\\egroup",
3288   format("\\wd\\mplibscratchbox %fbp", urx-llx),
3289   format("\\ht\\mplibscratchbox %fbp", ury-lly),
3290   "\\dp\\mplibscratchbox 0pt",
3291 })
3292 local grattr
3293 if trgroup.off then
3294   grattr = ""
3295 else
3296   local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>",
3297                                     trgroup.isolated, trgroup.knockout))
3298   grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3299 end
3300 local res = gather_resources("")
3301 local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3302 if pdfmode then
3303   if trgroup.wrapped then
3304     put2output(tableconcat{
3305       "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3306       "/BBox[" .. bbox .. "]} resources{" .. res .. "\\mplibscratchbox",
3307       "\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}",
3308     })
3309   end
3310   put2output(tableconcat{
3311     "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3312     "/BBox[" .. bbox .. "]", grattr, "} resources{" .. res .. "\\mplibscratchbox",
3313     "\\luamplibtagsgroupput{" .. trgroup.name .. "}" ..,
3314     [[\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}]],
3315     [[\\wd\\mplibscratchbox 0pt\\ht\\mplibscratchbox 0pt\\dp\\mplibscratchbox 0pt]],
3316     [[\\box\\mplibscratchbox]],
3317     "\\}\\endgroup",
3318     "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{",
3319     "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3320     "\\useboxresource \\the\\lastsavedboxresourceindex",

```

```

3321     "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3322     "\\box\\mplibscratchbox}",
3323 })
3324 else
3325     if trgroup.wrapped then
3326         trgroup.cnt = (trgroup.cnt or 0) + 1
3327         local objname = format("@mplibtrgr%s", trgroup.cnt)
3328         put2output(tableconcat{
3329             "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3330             "\\unhbox\\mplibscratchbox",
3331             "\\special{pdf:put @resources <<", res, ">>}",
3332             "\\special{pdf:exobj}",
3333             "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3334         })
3335     end
3336     trgroup.cnt = (trgroup.cnt or 0) + 1
3337     local objname = format("@mplibtrgr%s", trgroup.cnt)
3338     put2output(tableconcat{
3339         "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3340         "\\unhbox\\mplibscratchbox",
3341         "\\special{pdf:put @resources <<", res, ">>}",
3342         "\\special{pdf:exobj <<", grattr, ">>}",
3343         "\\luamplibtagasgroupput{",trgroup.name,"}{",
3344         "\\special{pdf:uxobj ", objname, "}",
3345         "}}\\endgroup",
3346     })
3347     token.set_macro("luamplib.group."..trgroup.name, tableconcat{
3348         "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
3349         "\\special{pdf:uxobj ", objname, "}",
3350         "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
3351         "\\box\\mplibscratchbox",
3352     }, "global")
3353 end
3354 trgroup.shifts[trgroup.name] = { llx, lly }
3355 end
3356 return grstate
3357 end
3358 function luamplib.registergroup (boxid, name, opts)
3359     if opts.asgroup and opts.asgroup:find"wrapped" then
3360         luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3361         run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
3362         opts.asgroup = opts.asgroup:gsub("wrapped","",)
3363     end
3364     local box = texgetbox(boxid)
3365     local wd, ht, dp = node.getwhd(box)
3366     local is_mask = opts.asgroup and opts.asgroup:find"masking"
3367     local res = opts.resources or ""
3368     res = gather_resources(res)
3369     local attr = { "/Type/XObject/Subtype/Form/FormType 1" }

```

```

3370 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
3371 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
3372 if opts.matrix and opts.matrix:find"%a" then
3373     local t = get_mp_matrix(opts.matrix)
3374     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3375 end
3376 local grtype = 3
3377 if opts.bbox then
3378     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3379     grtype = 2
3380 end
3381 local mpllx, mplly = get_macro'MPlLx', get_macro'MPlLy'
3382 if is_mask then
3383     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3384     t[5], t[6] = t[5]+mpllx, t[6]+mplly
3385     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3386     mpllx, mplly = 0, 0
3387 end
3388 if opts.matrix then
3389     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3390     grtype = opts.bbox and 4 or 1
3391 end
3392 if opts.asgroup and not opts.asgroup:find"off" then
3393     local t = { isolated = false, knockout = false, masking = false }
3394     for _,v in ipairs(opts.asgroup:gsub("%s",""):explode",+") do t[v] = true end
3395     local on
3396     if t.masking then
3397         on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorspace or "/DeviceGray"))
3398     else
3399         local cs = opts.colorspace and ("/CS%s"):format(opts.colorspace) or ""
3400         on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3401     end
3402     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3403 end
3404 local trgroup = pdfetcs.tr_group
3405 trgroup.shifts[name] = { mpllx, mplly }
3406 local whd
3407 if pdfmode then
3408     attr = tableconcat(attr) :gsub(decimals,rmzeros)
3409     local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3410     token.set_macro("luamplib.group."..name, tableconcat{
3411         "\\useboxresource ", index,
3412         }, "global")
3413     whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3414 else
3415     trgroup.cnt = (trgroup.cnt or 0) + 1
3416     local objname = format("@mplibtrgr%s", trgroup.cnt)
3417     texsprint (
3418         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",

```

```

3419     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3420     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3421     "\\special{pdf:bcontent}",
3422     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3423     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3424     "\\special{pdf:put @resources <<", res, ">>}",
3425     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3426     "\\special{pdf:econtent}}")
3427 )
3428 token.set_macro("luamplib.group.."name, tableconcat{
3429     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3430     "\\wd\\mplibscratchbox ", wd, "sp",
3431     "\\ht\\mplibscratchbox ", ht, "sp",
3432     "\\dp\\mplibscratchbox ", dp, "sp",
3433     "\\box\\mplibscratchbox",
3434     }, "global")
3435     whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3436 end
3437 info("w/h/d of group '%s': %s", name, whd)
3438 end
3439

```

luamplib.convert: flushing figures

```

3440 do
3441   local function stop_special_effects(fade,opaq)
3442     if opaq then -- opacity
3443       pdf_literalcode(opaq)
3444     end
3445     if fade then -- fading
3446       stop_pdf_code()
3447     end
3448   end
3449

```

For parsing prescript materials.

```

3450   local function script2table(s)
3451     local t = {}
3452     for _,i in ipairs(s:explode("\\13+")) do
3453       local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3454       if k and v and k ~= "" and not t[k] then
3455         t[k] = v
3456       end
3457     end
3458     return t
3459   end
3460

```

For path shading with transformed pen

```

3461   local function invert_matrix (t)
3462     local a, b, c, d, x, y = t[1], t[2], t[3], t[4], t[5], t[6]

```

```

3463     local det = a*d - b*c
3464     if det == 0 then err"transformation is not invertible!" end
3465     return format("%f %f %f %f %f %f cm ",
3466         d/det, 0-b/det, 0-c/det, a/det, (d*x-b*y)/det, (a*y-c*x)/det)
3467 end
3468

```

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

```

3469 local function pdf_textfigure(font,size,text,width,height,depth)
3470     text = text:gsub(".",function(c)
3471         return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
3472     end)
3473     put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
3474 end
3475
3476 local bend_tolerance = 131/65536
3477
3478 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3479
3480 local function pen_characteristics(object)
3481     local t = mplib.pen_info(object)
3482     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3483     divider = sx*sy - rx*ry
3484     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3485 end
3486
3487 local function concat(px, py) -- no tx, ty here
3488     return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
3489 end
3490
3491 local function curved(ith,pth)
3492     local d = pth.left_x - ith.right_x
3493     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3494         abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3495         d = pth.left_y - ith.right_y
3496         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3497             abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3498             return false
3499         end
3500     end
3501     return true
3502 end
3503
3504 local function flushnormalpath(path,open)
3505     local pth, ith
3506     for i=1,#path do
3507         pth = path[i]
3508         if not ith then

```

```

3509     pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3510 elseif curved(ith,pth) then
3511     pdf_literalcode("%f %f %f %f %f %f c",
3512         ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3513 else
3514     pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3515 end
3516 ith = pth
3517 end
3518 if not open then
3519     local one = path[1]
3520     if curved(pth,one) then
3521         pdf_literalcode("%f %f %f %f %f %f c",
3522             pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3523     else
3524         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3525     end
3526 elseif #path == 1 then -- special case .. draw point
3527     local one = path[1]
3528     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3529 end
3530 end
3531
3532 local function flushconcatpath(path,open)
3533     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3534     local pth, ith
3535     for i=1,#path do
3536         pth = path[i]
3537         if not ith then
3538             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3539         elseif curved(ith,pth) then
3540             local a, b = concat(ith.right_x,ith.right_y)
3541             local c, d = concat(pth.left_x,pth.left_y)
3542             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3543         else
3544             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3545         end
3546         ith = pth
3547     end
3548     if not open then
3549         local one = path[1]
3550         if curved(pth,one) then
3551             local a, b = concat(pth.right_x,pth.right_y)
3552             local c, d = concat(one.left_x,one.left_y)
3553             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3554         else
3555             pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3556         end
3557     elseif #path == 1 then -- special case .. draw point

```

```

3558     local one = path[1]
3559     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3560 end
3561 end
3562

```

Finally, flush figures by inserting PDF literals.

```

3563 local function flush (result,flusher)
3564   if result then
3565     local figures = result.fig
3566     if figures then
3567       for f=1, #figures do
3568         info("flushing figure %s",f)
3569         local figure = figures[f]
3570         local objects = figure:objects()
3571         local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3572         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3573         local bbox = figure:boundingbox()
3574         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3575         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()

3576   else

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

3577     if tex_code_pre_mplib[f] then
3578       put2output(tex_code_pre_mplib[f])
3579     end
3580     pdf_startfigure(fignum,llx,lly,urx,ury)
3581     start_pdf_code()
3582     if objects then
3583       local savedpath = nil
3584       local savedhtap = nil
3585       for o=1,#objects do
3586         local object      = objects[o]
3587         local objecttype  = object.type

```

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

```

3588     local prescript      = object.prescript
3589     prescript = prescript and script2table(prescript) -- prescript is now a table
3590     do_preobj_CR(object,prescript) -- color
3591     local fading_ = do_preobj_FADE(object,prescript) -- fading
3592     local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3593     do_preobj_PAT(object,prescript) -- tiling pattern
3594     do_preobj_shading(object,prescript) -- shading pattern

```

```

3595         local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3596     if prescript and prescript.mplibtexboxid then
3597         put_tex_boxes(object,prescript)
3598     elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3599     elseif objecttype == "start_clip" then
3600         local evenodd = not object.istext and object.postscript == "evenodd"
3601         start_pdf_code()
3602         flushnormalpath(object.path,false)
3603         pdf_literalcode(evenodd and "W* n" or "W n")
3604     elseif objecttype == "stop_clip" then
3605         stop_pdf_code()
3606         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3607     elseif objecttype == "special" then

```

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

```

3608         if prescript and prescript.postmplibverbtx then
3609             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
3610         end
3611     elseif objecttype == "text" then
3612         local ot = object.transform -- 3,4,5,6,1,2
3613         start_pdf_code()
3614         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3615         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3616         stop_pdf_code()
3617     elseif not trgroup and fading_ ~= "stop" then
3618         local evenodd, collect, both = false, false, false
3619         local postscript = object.postscript
3620         if not object.istext then
3621             if postscript == "evenodd" then
3622                 evenodd = true
3623             elseif postscript == "collect" then
3624                 collect = true
3625             elseif postscript == "both" then
3626                 both = true
3627             elseif postscript == "eoboth" then
3628                 evenodd = true
3629                 both = true
3630             end
3631         end
3632         if collect then
3633             if not savedpath then
3634                 savedpath = { object.path or false }
3635                 savedhtap = { object.htap or false }
3636             else
3637                 savedpath[#savedpath+1] = object.path or false
3638                 savedhtap[#savedhtap+1] = object.htap or false
3639             end
3640         else

```

Removed from ConTeXt general: color stuff.

```

3641         local ml = object.miterlimit
3642         if ml and ml ~= miterlimit then
3643             miterlimit = ml
3644             pdf_literalcode("%f M",ml)
3645         end
3646         local lj = object.linejoin
3647         if lj and lj ~= linejoin then
3648             linejoin = lj
3649             pdf_literalcode("%i j",lj)
3650         end
3651         local lc = object.linecap
3652         if lc and lc ~= linecap then
3653             linecap = lc
3654             pdf_literalcode("%i J",lc)
3655         end
3656         local dl = object.dash
3657         if dl then
3658             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3659             if d ~= dashed then
3660                 dashed = d
3661                 pdf_literalcode(dashed)
3662             end
3663         elseif dashed then
3664             pdf_literalcode("[ ] 0 d")
3665             dashed = false
3666         end
3667         local path = object.path
3668         local transformed, penwidth = false, 1
3669         local open = path and path[1].left_type and path[#path].right_type
3670         local pen = object.pen
3671         if pen then
3672             if pen.type == 'elliptical' then
3673                 transformed, penwidth = pen_characteristics(object) -- boolean, value

```

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

```

3674             if penwidth and pdfetcs.tr_group.widths then
3675                 tableinsert(pdfetcs.tr_group.widths, penwidth)
3676             end
3677             pdf_literalcode("%f w",penwidth)
3678             if objecttype == 'fill' then
3679                 objecttype = 'both'
3680             end
3681             else -- calculated by mplib itself
3682                 objecttype = 'fill'
3683             end
3684         end

```

Added : shading

```

3685         local shade_no, shade_stroke, shade_cm, shade_im = do_preobj_SH(object,prescript) -- shading
3686         if shade_no then

```

```

3687         objecttype = "shade"
3688         shade_im = transformed and invert_matrix{sx, rx, ry, sy, tx ,ty} or ""
3689         shade_cm = shade_cm and shade_cm.." cm " or ""
3690     end
3691     if transformed then
3692         start_pdf_code()
3693     end
3694     if path then
3695         if savedpath then
3696             for i=1,#savedpath do
3697                 local path = savedpath[i]
3698                 if transformed then
3699                     flushconcatpath(path,open)
3700                 else
3701                     flushnormalpath(path,open)
3702                 end
3703             end
3704             savedpath = nil
3705         end
3706         if transformed then
3707             flushconcatpath(path,open)
3708         else
3709             flushnormalpath(path,open)
3710         end
3711         if objecttype == "fill" then
3712             pdf_literalcode(evenodd and "h f*" or "h f")
3713         elseif objecttype == "outline" then
3714             if both then
3715                 pdf_literalcode(evenodd and "h B*" or "h B")
3716             else
3717                 pdf_literalcode(open and "S" or "h S")
3718             end
3719         elseif objecttype == "both" then
3720             pdf_literalcode(evenodd and "h B*" or "h B")
3721         elseif objecttype == "shade" then
3722             pdf_literalcode("q W%s %s %s%s/MPLibSh%s sh Q",
3723                 evenodd and "*" or "",
3724                 shade_stroke and "s" or "n",
3725                 shade_im, shade_cm, shade_no)
3726         end
3727     end
3728     if transformed then
3729         stop_pdf_code()
3730     end
3731     local path = object.htap

```

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

```

3732         if path then
3733             if transformed then

```

```

3734         start_pdf_code()
3735     end
3736     if savedhtap then
3737         for i=1,#savedhtap do
3738             local path = savedhtap[i]
3739             if transformed then
3740                 flushconcatpath(path,open)
3741             else
3742                 flushnormalpath(path,open)
3743             end
3744         end
3745         savedhtap = nil
3746         evenodd = true
3747     end
3748     if transformed then
3749         flushconcatpath(path,open)
3750     else
3751         flushnormalpath(path,open)
3752     end
3753     if objecttype == "fill" then
3754         pdf_literalcode(evenodd and "h f*" or "h f")
3755     elseif objecttype == "outline" then
3756         pdf_literalcode(open and "S" or "h S")
3757     elseif objecttype == "both" then
3758         pdf_literalcode(evenodd and "h B*" or "h B")
3759     elseif objecttype == "shade" then
3760         pdf_literalcode("q W%s %s %s%s/MPlibSh%s sh Q",
3761             evenodd and "*" or "",
3762             shade_stroke and "s" or "n",
3763             shade_im, shade_cm, shade_no)
3764     end
3765     if transformed then
3766         stop_pdf_code()
3767     end
3768 end
3769 end
3770 end

```

Added to ConT_EXt 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.

```

3771     if fading_ == "start" then
3772         stop_special_effects(false, tr_opaq)
3773         pdfetcs.fading.specialeffects = {fading_, false}
3774     elseif trgroup == "start" then

```

```

3775         pdfetcs.tr_group.specialeffects = {fading_, tr_opaq}
3776     elseif fading_ == "stop" then
3777         local se = pdfetcs.fading.specialeffects
3778         if se then stop_special_effects(se[1], se[2]) end
3779     elseif trgroup == "stop" then
3780         local se = pdfetcs.tr_group.specialeffects
3781         if se then stop_special_effects(se[1], se[2]) end
3782     else
3783         stop_special_effects(fading_, tr_opaq)
3784     end
3785     if fading_ or trgroup then -- extgs resetted
3786         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3787     end
3788 end
3789 end
3790 stop_pdf_code()
3791 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtex code.

```

3792     for _,v in ipairs(figcontents) do
3793         if type(v) == "table" then
3794             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3795         else
3796             texsprint(v)
3797         end
3798     end
3799     if #figcontents.post > 0 then texsprint(figcontents.post) end
3800     figcontents = { post = { } }
3801 end
3802 end
3803 end
3804 end
3805 end
3806
3807 function luamplib.convert (result, flusher)
3808     flush(result, flusher)
3809     return true -- done
3810 end
3811 end
3812
3813 function luamplib.colorconverter (cr)
3814     local n = #cr
3815     if n == 4 then
3816         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3817         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3818     elseif n == 3 then
3819         local r, g, b = cr[1], cr[2], cr[3]
3820         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3821     else

```

```

3822     local s = cr[1]
3823     return format("%.3f g %.3f G",s,s), "0 g 0 G"
3824 end
3825 end
3826

```

The code for externalize functionality

```

3827 do
3828   local extname, prefix, pdfname, luaname, figtab, prevfigtab
3829   local majorV, minorV = pdf.getmajorversion(), pdf.getminorversion()
3830   local function is_shell_esc()
3831     if status.shell_escape == 1 then return true end
3832     luatexbase.module_error("luamplib",
3833       "--shell-escape is needed for 'externalize' feature.\n\z
3834       Rerun with --shell-escape option.")
3835   end
3836
3837   function externalize.setup ()
3838     extname = externalize.MY_NAME or format("%s-mplib-figure",tex.jobname)
3839     externalize.running = true
3840     local dir = status.output_directory or "."
3841     prefix = format("%s/%s",dir,extname)
3842     luaname = format("%ss.lua",prefix,extname)
3843     pdfname = format("%s/%s.pdf",dir,tex.jobname)
3844
3845     local extver = format("20260922.%s%s", majorV, minorV)
3846     figtab = { version = extver }
3847     if lfs.isfile(luaname) then
3848       prevfigtab = require(luaname)
3849     end
3850     if not prevfigtab or prevfigtab.version ~= extver then
3851       prevfigtab = { }
3852       externalize.activate()
3853     end
3854
3855     luatexbase.add_to_callback("finish_pdffile", function()
3856       local mandatory = figtab.mandatory
3857       if mandatory then
3858         local depends = externalize.depends
3859         for i = 1, #figtab do
3860           if mandatory[i] then
3861             for _,v in ipairs(depends[i]) do
3862               mandatory[v] = true
3863             end
3864           end
3865         end
3866       end
3867       if not figtab.found then
3868         table.tofile(luaname, figtab, "return")

```

```

3869         if not mandatory or not is_shell_esc() then return end
3870         luatexbase.add_to_callback("wrapup_run", function()
3871             os.exec{arg[0], "--halt-on-error", tableunpack(arg)}
3872         end, "luamplib_externalize")
3873     end
3874 end,
3875 "luamplib_externalize")
3876 end
3877 function externalize.activate ()
3878     figtab.active = true
3879     texsprint"\AddToHook{shipout/before}\z
3880     {\directlua{luamplib.externalize.shipout(tex.getbox(\the\ShipoutBox))}}"
3881 end
3882 function externalize.latelua(wd,ht,dp,mgn)
3883     local llx, lly = pdf.getpos()
3884     local urx, ury = (llx+wd+mgn)/factor, (lly+ht+mgn)/factor
3885     llx, lly = (llx-mgn)/factor, (lly-dp-mgn)/factor
3886     local cropbox = ("/CropBox[%f %f %f %f]"):format(llx, lly, urx, ury)
3887     pdf.setpageattributes(cropbox)
3888 end
3889 function externalize.shipout (head)
3890     local attr = luatexbase.attributes.luamplibexternalizeattr
3891     local getnext, has_attribute = node.getnext, node.has_attribute
3892     local curr = head
3893     while curr do
3894         if curr.head then
3895             local count = attr and has_attribute(curr,attr)
3896             if count then
3897                 local page = (externalize.page or tex.count.ReadonlyShipoutCounter) + 1
3898                 externalize.page = page
3899
3900                 local fig = figtab[count]
3901                 local wd, ht, dp = curr.width, curr.height, curr.depth
3902
3903                 local latelua = node.new("whatsit","late_lua")
3904                 latelua.token = format("luamplib.externalize.latelua(%s,%s,%s,%s)",wd,ht,dp,fig.margin or 0)
3905                 latelua.name = "luamplib.externalize.latelua"
3906                 curr.head = node.insert_before(curr.head, curr.head, latelua)
3907
3908                 fig.pages = fig.pages or { }
3909                 tableinsert(fig.pages, page)
3910                 fig.metric = fig.metric or { }
3911                 tableinsert(fig.metric, { wd, ht, dp })
3912
3913                 tex.setbox("mplibscratchbox",node.copy(curr))
3914                 tex.shipout"mplibscratchbox"
3915
3916                 figtab.found = true
3917             else

```

```

3918         externalize.shipout(curr.head)
3919     end
3920     elseif curr.leader and curr.leader.head then
3921         externalize.shipout(curr.leader.head)
3922     end
3923     curr = getnext(curr)
3924 end
3925
3926 if figtab.found then
3927     if not luatexbase.in_callback("wrapup_run","luamplib_externalize") then
3928         if is_shell_esc() then
3929             externalize.wrapup_run()
3930         else
3931             luatexbase.add_to_callback("wrapup_run",function() end,"luamplib_externalize")
3932         end
3933     end
3934     texsprint"\DiscardShipoutBox"
3935 end
3936 end
3937 function externalize.wrapup_run ()
3938     luatexbase.add_to_callback("wrapup_run", function()
3939         table.tofile(luaname, figtab, "return")
3940
3941         for i,v in ipairs(figtab) do
3942             if v.pages then
3943                 local mgn = v.margin and v.margin*2 or 0
3944                 for ii,vv in ipairs(v.pages) do
3945                     local wd, ht, dp = tableunpack(v.metric[ii])
3946                     local status = os.spawn(format(
3947                         "luatex --halt-on-error --jobname=%s-%s-%s \z
3948                         \"\pagewidth=%ssp\pageheight=%ssp\z
3949                         \pdfvariable horigin 0pt\pdfvariable vorigin 0pt\z
3950                         \pdfvariable majorversion %s\pdfvariable minorversion %s\z
3951                         \topskip=0pt\nopagenumbers\z
3952                         \directlua{img.write{filename=[[[%s]],page=%s,pagebox=[[crop]]]}\bye\"",
3953                         extname, i, ii, wd+mgn, ht+dp+mgn, majorV, minorV, pdfname, vv))
3954                     assert(status == 0, format("failed to generate external image No. %s!",i))
3955                     os.remove(format("%s-%s-%s.log", prefix, i, ii))
3956                 end
3957             end
3958         end
3959
3960         os.exec{arg[0], "--halt-on-error", tableunpack(arg)}
3961     end,
3962     "luamplib_externalize")
3963 end
3964 function externalize.figure(data)
3965     local count = tex.count.luamplibexternalizecount + 1
3966     tex.setcount("global", "luamplibexternalizecount", count)

```

```

3967
3968     local prevfig = prevfigtab[count]
3969     local num_of_figs, whd
3970     if prevfig then
3971         num_of_figs = prevfig.pages and #prevfig.pages
3972                     or prevfig.changed and 0
3973                     or prevfig.num_of_figs
3974         whd = prevfig.metric or prevfig.whd
3975     end
3976     local depend = get_macro"luamplibexternalizedependson"
3977
3978     local margin = get_macro"luamplibexternalizemargin"
3979     margin = margin and tex.sp(margin)
3980
3981     figtab[count] = {
3982         data = data,
3983         depend = depend,
3984         margin = margin,
3985         num_of_figs = num_of_figs,
3986         whd = whd,
3987     }
3988
3989     if get_macro"luamplibexternalizedothis" == "false" then return end
3990
3991     local match = prevfig and prevfig.data == data
3992                 and prevfig.depend == depend
3993                 and prevfig.margin == margin
3994
3995     if prevfigtab.mandatory and prevfigtab.mandatory[count] then
3996         goto do_this_fig
3997     end
3998     if depend then
3999         local deps = depend:explode","
4000         for i,v in ipairs(deps) do
4001             local n = tonumber(v)
4002             if n < 0 then
4003                 n = count + n
4004             end
4005             deps[i] = n
4006         end
4007
4008         local depends = externalize.depends or { }
4009         externalize.depends = depends
4010         depends[count] = deps
4011         for _,v in ipairs(deps) do
4012             depends[v] = depends[v] or { }
4013             tableinsert(depends[v], count)
4014         end
4015

```

```

4016     local done = 0
4017     for _,v in ipairs(deps) do
4018         if figtab[v].changed then
4019             done = done + 1
4020         end
4021     end
4022
4023     if done == #deps then -- all changed: OK
4024         goto do_this_fig
4025     elseif done == 0 and match then -- none changed and match: OK
4026     else
4027         match = true -- avoid error
4028         figtab.mandatory = figtab.mandatory or { }
4029         for _,v in ipairs(deps) do
4030             figtab.mandatory[v] = true
4031         end
4032     end
4033 end
4034
4035 if match then
4036     if not num_of_figs then goto do_this_fig end
4037     for i = 1, num_of_figs do
4038         local name = format("%s-%s-%s.pdf", prefix, count, i)
4039         if not lfs.isfile(name) then goto do_this_fig end
4040         local wd, ht, dp = tableunpack(whd[i])
4041         texsprint(ccexplat,
4042             "\\prependtombibbox\\hbox dir TLT\\bgroup",
4043             "\\tag_socket_use:nn{luamplib/figure/begin}\\l__luamplib_tag_alt_dflt_tl",
4044             "\\setbox\\mplibscratchbox\\vbox to", ht+dp, "sp{\\vss\\hbox to", wd, "sp{\\hss",
4045             "\\directlua{img.write{filename=[[", name, "]]}}\\hss}\\vss}",
4046             "\\dp\\mplibscratchbox=", dp, "sp",
4047             "\\ht\\mplibscratchbox=\\dimexpr\\ht\\mplibscratchbox-\\dp\\mplibscratchbox\\relax",
4048             "\\tag_socket_use:nnn{luamplib/figure/end}\\mplibscratchbox}\\box\\mplibscratchbox}",
4049             "\\egroup")
4050     end
4051     return true
4052 end
4053
4054 ::do_this_fig::
4055 if not figtab.active then
4056     externalize.activate()
4057 end
4058 figtab[count].changed = true
4059 texsprint "\\luamplibexternalizethisfigure"
4060 end
4061 local index = luatexbase.new_luafunction "luamplib_externalize"
4062 lua.get_functions_table()[index] = function()
4063     local data = token.scan_argument()
4064     if externalize.running and externalize.figure(data) then

```

```

4065     tex.setcount("l_tmpa_int", 1)
4066   else
4067     tex.setcount("l_tmpa_int", 0)
4068   end
4069 end
4070 token.set_lua("luamplib@externalized", index, "global")
4071 end

```

2.2 T_EX package

First we need to load some packages.

```

4072 \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.

```

4073 \NeedsTeXFormat{LaTeX2e}
4074 \ProvidesPackage{luamplib}
4075 [2026/09/22 v2.44.0 mplib package for LuaTeX]
4076 \fi
4077 \ifdefined\newluafunction\else
4078 \input ltluatex
4079 \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.

```

4080 \ifnum\outputmode=0
4081 \ifdefined\AddToHookNext
4082 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
4083 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
4084 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
4085 \else
4086 \input atbegshi.sty
4087 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
4088 \let\luamplibatfirstshipout\AtBeginShipoutFirst
4089 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
4090 \fi
4091 \fi

```

Loading of lua code.

```

4092 \directlua{require("luamplib")}

```

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

```

4093 \ifx\pdfoutput\undefined
4094 \let\pdfoutput\outputmode
4095 \fi
4096 \ifx\pdfliteral\undefined
4097 \protected\def\pdfliteral{\pdfextension literal}
4098 \fi

```

Set the format for METAPOST.

```
4099 \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.

```
4100 \ifnum\pdfoutput>0
4101   \let\mplibtoPDF\pdfliteral
4102 \else
4103   \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
4104   \ifcsname PackageInfo\endcsname
4105     \PackageInfo{luamplib}{only dvipdfmx is supported currently}
4106   \else
4107     \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
4108   \fi
4109 \fi
```

To make mplibcode typeset always in horizontal mode.

```
4110 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
4111 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
4112 \mplibnoforcehmode
```

Catcode. We want to allow comment sign in mplibcode.

```
4113 \def\mplibsetupcatcodes{%
4114   %catcode`\{=12 %catcode`\}=12
4115   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
4116   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
4117 }
```

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

```
4118 \ifnum\outputmode>0 %
4119   \def\mplibputtextbox#1#2#3#4{%
4120     \pdfextension save\relax
4121     \vbox to 0pt{\vss
4122       \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
4123       \kern#3bp}%
4124     \pdfextension restore\relax}
4125 \else
4126   \def\mplibputtextbox#1#2#3#4{%
4127     \special{pdf:btrans matrix #4 #2 #3}%
4128     \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
4129     \special{pdf:etrans}}
4130 \fi
```

use Transparency Group

```
4131 \protected\def\usemplibgroup#1#2{\usemplibgroupmain}
4132 \def\usemplibgroupmain#1{%
4133   \prependtomplibbox\hbox dir TLT\bgroup
4134   \csname luamplib.group.#1\endcsname
4135   \egroup
4136 }
```

```

4137 \protected\def\mplibgroup#1{%
4138   \begingroup
4139   \def\MPllx{0}\def\MPlly{0}%
4140   \def\mplibgroupname{#1}%
4141   \mplibgroupgetnexttok
4142 }
4143 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
4144 \def\mplibgroupskipsspace{\afterassignment\mplibgroupgetnexttok\let\nexttok= }
4145 \def\mplibgroupbranch{%
4146   \ifx [\nexttok
4147     \expandafter\mplibgroupopts
4148   \else
4149     \ifx\mplibsptoken\nexttok
4150       \expandafter\expandafter\expandafter\mplibgroupskipsspace
4151     \else
4152       \let\mplibgroupoptions\empty
4153       \expandafter\expandafter\expandafter\mplibgroupmain
4154     \fi
4155   \fi
4156 }
4157 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
4158 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
4159 \protected\def\endmplibgroup{\egroup
4160   \directlua{ luamplib.registergroup(
4161     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
4162   )}%
4163   \endgroup
4164 }

```

Patterns

```

4165 {\def\:{\global\let\mplibsptoken= } \: }
4166 \protected\def\mplibpattern#1{%
4167   \begingroup
4168   \def\MPllx{0}\def\MPlly{0}%
4169   \def\mplibpatternname{#1}%
4170   \mplibpatterngetnexttok
4171 }
4172 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
4173 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
4174 \def\mplibpatternbranch{%
4175   \ifx [\nexttok
4176     \expandafter\mplibpatternopts
4177   \else
4178     \ifx\mplibsptoken\nexttok
4179       \expandafter\expandafter\expandafter\mplibpatternskipsspace
4180     \else
4181       \let\mplibpatternoptions\empty
4182       \expandafter\expandafter\expandafter\mplibpatternmain
4183     \fi

```

```

4184 \fi
4185 }
4186 \def\mplibpatternopts[#1]{%
4187 \def\mplibpatternoptions{#1}%
4188 \mplibpatternmain
4189 }
4190 \def\mplibpatternmain{%
4191 \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
4192 }
4193 \protected\def\endmpfigpattern{%
4194 \egroup
4195 \directlua{ luamplib.registerpattern(
4196 \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
4197 )}%
4198 \endgroup
4199 }

```

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

```

4200 \def\mpfiginstancename{@mpfig}
4201 \protected\def\mpfig{%
4202 \begingroup
4203 \futurelet\nexttok\mplibmpfigbranch
4204 }
4205 \def\mplibmpfigbranch{%
4206 \ifx *\nexttok
4207 \expandafter\mplibprempfig
4208 \else
4209 \ifx [\nexttok
4210 \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
4211 \else
4212 \expandafter\expandafter\expandafter\mplibmainmpfig
4213 \fi
4214 \fi
4215 }
4216 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
4217 \def\mplibmainmpfig{%
4218 \begingroup
4219 \mplibsetupcatcodes
4220 \mplibdomainmpfig
4221 }
4222 \long\def\mplibdomainmpfig#1\endmpfig{%
4223 \endgroup
4224 \directlua{
4225 local legacy = luamplib.legacyverbatimimtx
4226 local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
4227 local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
4228 luamplib.legacyverbatimimtx = false
4229 luamplib.everymplib["\mpfiginstancename"] = ""
4230 luamplib.everyendmplib["\mpfiginstancename"] = ""

```

```

4231   luamplib.process_mplibcode(
4232     "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===".." "..everyendmpfig.." endfig;",
4233     "\mpfiginstancename")
4234   luamplib.legacyverbatimtex = legacy
4235   luamplib.everymplib["\mpfiginstancename"] = everympfig
4236   luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
4237 }%
4238 \endgroup
4239 }
4240 \def\mplibprempfig#1{%
4241   \begingroup
4242   \mplibsetupcatcodes
4243   \mplibdoprempfig
4244 }
4245 \long\def\mplibdoprempfig#1\endmpfig{%
4246   \endgroup
4247   \directlua{
4248     local legacy = luamplib.legacyverbatimtex
4249     local everympfig = luamplib.everymplib["\mpfiginstancename"]
4250     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
4251     luamplib.legacyverbatimtex = false
4252     luamplib.everymplib["\mpfiginstancename"] = ""
4253     luamplib.everyendmplib["\mpfiginstancename"] = ""
4254     luamplib.process_mplibcode([===[\unexpanded{#1}]==="..\mpfiginstancename")
4255     luamplib.legacyverbatimtex = legacy
4256     luamplib.everymplib["\mpfiginstancename"] = everympfig
4257     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
4258   }%
4259   \endgroup
4260 }
4261 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

4262 \unless\ifcsname ver@luamplib.sty\endcsname
4263   \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
4264   \protected\def\mplibcode{%
4265     \begingroup
4266     \futurelet\nexttok\mplibcodebranch
4267   }
4268   \def\mplibcodebranch{%
4269     \ifx [\nexttok
4270       \expandafter\mplibcodegetinstancename
4271     \else
4272       \global\let\currentmpinstancename\empty
4273       \expandafter\mplibcodeindeed
4274     \fi
4275   }
4276   \def\mplibcodeindeed{%
4277     \begingroup

```

```

4278 \mplibsetupcatcodes
4279 \mplibdocode
4280 }
4281 \long\def\mplibdocode#1\endmplibcode{%
4282 \endgroup
4283 \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]==],"\currentmpinstancename")}%
4284 \endgroup
4285 }
4286 \protected\def\endmplibcode{endmplibcode}
4287 \else

```

The $\LaTeX$ -specific part: a new environment.

```

4288 \newenvironment{mplibcode}[1][{}]{%
4289 \xdef\currentmpinstancename{#1}%
4290 \mplibtmptoks{}\ltxdomplibcode
4291 }{}
4292 \def\ltxdomplibcode{%
4293 \begingroup
4294 \mplibsetupcatcodes
4295 \ltxdomplibcodeindeed
4296 }
4297 \def\mplib@mplibcode{mplibcode}
4298 \long\def\ltxdomplibcodeindeed#1\end#2{%
4299 \endgroup
4300 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
4301 \def\mplibtemp@a{#2}%
4302 \ifx\mplib@mplibcode\mplibtemp@a
4303 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]==],"\currentmpinstancename")}%
4304 \end{mplibcode}%
4305 \else
4306 \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
4307 \expandafter\ltxdomplibcode
4308 \fi
4309 }
4310 \fi

```

User settings.

```

4311 \def\mplibshowlog#1{\directlua{
4312 local s = string.lower("#1")
4313 if s == "enable" or s == "true" or s == "yes" then
4314 luamplib.showlog = true
4315 else
4316 luamplib.showlog = false
4317 end
4318 }}
4319 \def\mpliblegacybehavior#1{\directlua{
4320 local s = string.lower("#1")
4321 if s == "enable" or s == "true" or s == "yes" then
4322 luamplib.legacyverbatim = true
4323 else

```

```

4324     luamplib.legacyverbatimex = false
4325   end
4326 }}
4327 \def\mplibverbatim#1{\directlua{
4328   local s = string.lower("#1")
4329   if s == "enable" or s == "true" or s == "yes" then
4330     luamplib.verbatiminput = true
4331   else
4332     luamplib.verbatiminput = false
4333   end
4334 }}
4335 \newtoks\mplibtmptoks

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

4336 \ifcsname ver@luamplib.sty\endcsname
4337   \protected\def\everymplib{%
4338     \begingroup
4339     \mplibsetupcatcodes
4340     \mplibdoeverymplib
4341   }
4342   \protected\def\everyendmplib{%
4343     \begingroup
4344     \mplibsetupcatcodes
4345     \mplibdoeveryendmplib
4346   }
4347   \newcommand\mplibdoeverymplib[2][]{%
4348     \endgroup
4349     \directlua{
4350       luamplib.everymplib["#1"] = [===[\unexpanded{#2}]===]
4351     }%
4352   }
4353   \newcommand\mplibdoeveryendmplib[2][]{%
4354     \endgroup
4355     \directlua{
4356       luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===]
4357     }%
4358   }
4359 \else
4360   \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
4361   \protected\def\everymplib#1{%
4362     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4363     \begingroup
4364     \mplibsetupcatcodes
4365     \mplibdoeverymplib
4366   }
4367   \long\def\mplibdoeverymplib#1{%
4368     \endgroup
4369     \directlua{
4370       luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]

```

```

4371 }%
4372 }
4373 \protected\def\everyendmplib#1#{%
4374 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4375 \begingroup
4376 \mplibsetupcatcodes
4377 \mplibdoeveryendmplib
4378 }
4379 \long\def\mplibdoeveryendmplib#1{%
4380 \endgroup
4381 \directlua{
4382     luaplib.everyendmplib["\currentmpinstancename"] = [==[\unexpanded{#1}]==]
4383 }%
4384 }
4385 \fi

```

TeX macros for dimen/color

```

4386 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
4387 \def\mpcolor#1#{\domplibcolor{#1}}
4388 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

4389 \def\mplibnumbersystem#1{\directlua{
4390     local t = "#1"
4391     if t == "binary" then t = "decimal" end
4392     luaplib.numbersystem = t
4393 }}

```

Settings for .mp cache files.

```

4394 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4395 \def\mplibdomakenocache#1,{%
4396 \ifx\empty#1\empty
4397 \expandafter\mplibdomakenocache
4398 \else
4399 \ifx\stop#1\else
4400 \directlua{luaplib.noneedtoreplace["#1.mp"]=true}%
4401 \expandafter\expandafter\expandafter\mplibdomakenocache
4402 \fi
4403 \fi
4404 }
4405 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4406 \def\mplibdocancelnocache#1,{%
4407 \ifx\empty#1\empty
4408 \expandafter\mplibdocancelnocache
4409 \else
4410 \ifx\stop#1\else
4411 \directlua{luaplib.noneedtoreplace["#1.mp"]=false}%
4412 \expandafter\expandafter\expandafter\mplibdocancelnocache
4413 \fi
4414 \fi

```

```

4415 }
4416 \def\mplibcachedir#1{\directlua{luampplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4417 \def\mplibtexttextlabel#1{\directlua{
4418   local s = string.lower("#1")
4419   if s == "enable" or s == "true" or s == "yes" then
4420     luampplib.texttextlabel = true
4421   else
4422     luampplib.texttextlabel = false
4423   end
4424 }}
4425 \def\mplibcodeinherit#1{\directlua{
4426   local s = string.lower("#1")
4427   if s == "enable" or s == "true" or s == "yes" then
4428     luampplib.codeinherit = true
4429   else
4430     luampplib.codeinherit = false
4431   end
4432 }}
4433 \def\mplibglobaltexttext#1{\directlua{
4434   local s = string.lower("#1")
4435   if s == "enable" or s == "true" or s == "yes" then
4436     luampplib.globaltexttext = true
4437   else
4438     luampplib.globaltexttext = false
4439   end
4440 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

4441 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4442 \def\mplibstarttoPDF#1#2#3#4{%
4443   \prependtomplibbox
4444   \hbox dir TLT \luamplibexternalizemaybe \bgroup
4445   \xdef\MPllx{#1}\xdef\MPlly{#2}%
4446   \xdef\MPurx{#3}\xdef\MPury{#4}%
4447   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4448   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4449   \parskip0pt%
4450   \leftskip0pt%
4451   \parindent0pt%
4452   \everypar{}%
4453   \setbox\mplibscratchbox\vbox\bgroup
4454   \noindent
4455 }
4456 \def\mplibstoptoPDF{%
4457   \par

```

```

4458 \egroup %
4459 \setbox\mplibscratchbox\hbox %
4460   {\hskip-\MPllx bp%
4461    \raise-\MPlly bp%
4462    \box\mplibscratchbox}%
4463 \setbox\mplibscratchbox\vbox to \MPheight
4464   {\vfill
4465    \hsize\MPwidth
4466    \wd\mplibscratchbox\opt%
4467    \ht\mplibscratchbox\opt%
4468    \dp\mplibscratchbox\opt%
4469    \box\mplibscratchbox}%
4470 \wd\mplibscratchbox\MPwidth
4471 \ht\mplibscratchbox\MPheight
4472 \box\mplibscratchbox
4473 \egroup
4474 }

```

Text items have a special handler.

```

4475 \def\mplibtexttext#1#2#3#4#5{%
4476   \begingroup
4477   \setbox\mplibscratchbox\hbox
4478     {\font\temp=#1 at #2bp%
4479     \temp
4480     #3}%
4481   \setbox\mplibscratchbox\hbox
4482     {\hskip#4 bp%
4483     \raise#5 bp%
4484     \box\mplibscratchbox}%
4485   \wd\mplibscratchbox\opt%
4486   \ht\mplibscratchbox\opt%
4487   \dp\mplibscratchbox\opt%
4488   \box\mplibscratchbox
4489   \endgroup
4490 }

```

Input `luamplib.cfg` when it exists.

```

4491 \openin0=luamplib.cfg
4492 \ifeof0 \else
4493   \closein0
4494   \input luamplib.cfg
4495 \fi

```

Code for $\LaTeX$, especially `tagpdf`

```

4496 \def\luamplibtagtextboxset#1#2{#2}
4497 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4498 \let\luamplibtagasgroupset\relax
4499 \let\luamplibtagasgroupput\luamplibtagtextboxset
4500 \let\luamplibexternalizemaybe\relax
4501 \ifcsname SuspendTagging\endcsname\else\endinput\fi

```

4502

externalize functionality supports L^AT_EX PDF mode only.

```

4503 \def\mplibexternalize{%
4504   \newcount\luamplibexternalizecount
4505   \newattribute\luamplibexternalizeattr
4506   \chardef\luamplibexternalizealloc\allocationnumber
4507   \def\luamplibexternalizethisfigure{%
4508     \def\luamplibexternalizemaybe{attr \luamplibexternalizealloc =\luamplibexternalizecount}%
4509   }%
4510   \directlua{luamplib.externalize.setup()}%
4511   \def\luamplibexternalizedothis{true}%
4512   \aftergroup\mplibexternalizesuspend
4513 }
4514 \def\mplibexternalizename#1{\directlua{luamplib.externalize.MY_NAME=[===[#1]===]}}
4515 \def\mplibexternalizesuspend{\ifdefined\luamplibexternalizecount
4516   \directlua{luamplib.externalize.running = false}\fi}
4517 \def\mplibexternalizeresume{\ifdefined\luamplibexternalizecount
4518   \directlua{luamplib.externalize.running = true}\fi}
4519
4520
4521 \ifcsname ver@tagpdf.sty\endcsname \else
4522   \ExplSyntaxOn
4523   \keys_define:nn{luamplib/tagging}
4524   {
4525     ,alt          .code:n = { }
4526     ,actualtext   .code:n = { }
4527     ,artifact     .code:n = { }
4528     ,text         .code:n = { }
4529     ,off          .code:n = { }
4530     ,tag          .code:n = { }
4531     ,adjust-BBox  .code:n = { }
4532     ,tagging-setup .code:n = { }
4533     ,externalize  .code:n = { \cs_set_nopar:Npn \luamplibexternalizedothis {#1} }
4534     ,externalize  .default:n = { true }
4535     ,externalizedependson .code:n = { \cs_set_nopar:Npn \luamplibexternalizedependson {#1} }
4536     ,externalizemargin .code:n = { \cs_set_nopar:Npn \luamplibexternalizemargin {#1} }
4537     ,instance     .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4538     ,instancename .meta:n = { instance = {#1} }
4539     ,unknown      .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4540   }
4541   \RenewDocumentCommand\mplibcode{O{}}
4542   {
4543     \tl_gclear:N \currentmpinstancename
4544     \keys_set:ne{luamplib/tagging}{#1}
4545     \mplibtmptoks{}\ltxdomplibcode
4546   }
4547   \RenewDocumentCommand\mpfig{s O{}}
4548   {

```

```

4549 \group_begin:
4550 \keys_set_known:ne {luamplib/tagging} {#2}
4551 \IfBooleanTF{#1} { \mplibprempfig * }
4552           { \mplibmainmpfig }
4553 }
4554 \RenewDocumentCommand\usemplibgroup{0{ } m}
4555 {
4556   \group_begin:
4557   \keys_set_known:ne {luamplib/tagging} {#1}
4558   \luamplib@externalized{luamplib.group.#2}
4559   \int_compare:nNnT {\l_tmpa_int} = {\c_zero_int}
4560   {
4561     \prependtomplibox\hbox dir~TLT~\luamplibexternalizemaybe \bgroup
4562     \csname luamplib.group.#2\endcsname
4563     \egroup
4564   }
4565   \group_end:
4566 }
4567 \cs_set_eq:NN \mplibaltext \use_none:n
4568 \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 LATEX.
Why we used vbox initially? where hbox seems to be sufficient. Anyway, among various solu-
tions including \partokencontext\z@, \let\par\@@par, and \endgraf, we here attempt to address
the issue by adding the following line, which LATEX's \everypar should have done.

4569 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4570 \ExplSyntaxOff
4571 \endinput\fi
4572 \ExplSyntaxOn
4573 \tl_new:N \l__luamplib_tag_envname_tl
4574 \tl_new:N \l__luamplib_tag_alt_tl
4575 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4576 \tl_new:N \l__luamplib_tag_actual_tl
4577 \tl_new:N \l__luamplib_tag_struct_tl
4578 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4579 \bool_new:N \l__luamplib_tag_usetext_bool
4580 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4581 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4582 \tl_new:N \l__luamplib_tag_bboxdraw_tl
4583 \tl_new:N \l__luamplib_BBox_llx_tl
4584 \tl_new:N \l__luamplib_BBox_lly_tl
4585 \tl_new:N \l__luamplib_BBox_urx_tl
4586 \tl_new:N \l__luamplib_BBox_ury_tl
4587 \msg_new:nnn {luamplib}{figure-text-reuse}
4588 {
4589   tex-text~box~#1~probably~is~incorrectly~tagged.~
4590   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4591   Check~the~resulting~PDF.

```

```

4592 }
4593 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4594 {
4595   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4596   Using~mplibgroup~with~text~mode~is~not~recommended.~
4597   Check~the~resulting~PDF.
4598 }
4599 \msg_new:nnn{luamplib}{alt-text-missing}
4600 {
4601   Alternate~text~for~#1~is~missing.~
4602   Using~the~default~value~'#2'~instead.
4603 }

```

Sockets for tex-text boxes.

```

4604 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4605 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4606 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4607 {

```

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.

```

4608   \bool_if:NTF \l__luamplib_tag_usetext_bool
4609   {
4610     \tag_mc_end_push:
4611     \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4612     \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.

```

4613     \tag_mc_begin:n{tag=text}
4614     #2
4615     \tag_mc_end:
4616     \tag_struct_end:
4617     \tag_mc_begin_pop:n{ }
4618   }
4619   {
4620     \tag_suspend:n{\luamplibtagtextboxset}
4621     #2
4622     \tag_resume:n{\luamplibtagtextboxset}
4623   }
4624 }
4625 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4626 {
4627   \bool_lazy_and:nnTF
4628   { \l__luamplib_tag_usetext_bool }
4629   { \cs_if_free_p:c {luamplib.notaggedbox.#1} }
4630   {
4631     \tag_resume:n{\mplibputtextbox}
4632     \tag_mc_end:
4633     \cs_if_exist:cTF {luamplib.taggedbox.#1}
4634     {

```

```

4635 \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4636 #2
4637 \cs_undefine:c {luamplib.taggedbox.#1}
4638 }
4639 {
4640 \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
4641 \tag_mc_begin:n{ }
4642 \int_set:Nn \l_tmpa_int {#1}
4643 \tag_mc_reset_box:N \l_tmpa_int
4644 #2
4645 \tag_mc_end:
4646 }
4647 \tag_mc_begin:n{artifact}
4648 }
4649 {
4650 \int_set:Nn \l_tmpa_int {#1}
4651 \tag_mc_reset_box:N \l_tmpa_int
4652 #2
4653 }
4654 }
4655 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4656 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4657 \cs_set_nopar:Npn \luamplibtagtextboxset
4658 {
4659 \tag_socket_use:nnn{luamplib/texttext/set}
4660 }

```

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

```

4661 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4662 {
4663 \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
4664 \bool_set_false:N \l__luamplib_tag_usetext_bool
4665 \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
4666 \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4667 \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4668 }
4669 \sys_if_output_pdf:TF
4670 {
4671 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4672 {
4673 \pdfextension save\relax
4674 \vbox to 0pt{\vss
4675 \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4676 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4677 \kern #3bp}
4678 \pdfextension restore\relax
4679 }
4680 }

```

```

4681 {
4682   \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4683   {
4684     \special{pdf:btrans~matrix~#4~#2~#3}
4685     \vbox to 0pt{\vss\hbox bdir0 to 0pt{
4686       \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4687     \special{pdf:etrans}
4688   }
4689 }

```

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

```

4690 \cs_set_nopar:Npn \luamplibtagasgroupset
4691 {
4692   \bool_set_false:N \l__luamplib_tag_usetext_bool
4693 }
4694 \cs_set_nopar:Npn \luamplibtagasgroupput
4695 {
4696   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4697   \tag_socket_use:nnn{luamplib/mplibgroup/put}
4698 }

```

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

```

4699 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4700 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4701 {
4702   \cs_if_free:cT {luamplib.mplibgroup.text.#1}
4703   {
4704     \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4705     \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4706   }
4707   \tag_mc_end:
4708   \tag_mc_begin:n{tag=text}
4709   #2
4710   \tag_mc_end:
4711   \tag_mc_begin:n{artifact}
4712 }
4713 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

A macro for BBox attribute

```

4714 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4715 {
4716   \tl_set:Ne \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4717   \tex_savepos:D
4718   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4719   \tl_set:Ne \l__luamplib_BBox_llx_tl
4720   { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4721   \tl_set:Ne \l__luamplib_BBox_lly_tl
4722   { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4723   \tl_set:Ne \l__luamplib_BBox_urx_tl

```

```

4724 { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4725 \tl_set:Ne \l__luamplib_BBox_ury_tl
4726 { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4727 \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4728 {
4729   \int_zero:N \l_tmpa_int
4730   \tl_map_inline:nn
4731   {
4732     \l__luamplib_BBox_llx_tl
4733     \l__luamplib_BBox_lly_tl
4734     \l__luamplib_BBox_urx_tl
4735     \l__luamplib_BBox_ury_tl
4736   }
4737   {
4738     \int_incr:N \l_tmpa_int
4739     \tl_set:Ne ##1
4740     {
4741       \fp_eval:n
4742       {
4743         ##1
4744         +
4745         \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4746       }
4747     }
4748   }
4749 }
4750 \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4751 {
4752   /O /Layout /BBox [
4753     \l__luamplib_BBox_llx_tl\c_space_tl
4754     \l__luamplib_BBox_lly_tl\c_space_tl
4755     \l__luamplib_BBox_urx_tl\c_space_tl
4756     \l__luamplib_BBox_ury_tl
4757   ]
4758 }
4759 \bool_if:NT \l__tag_graphic_debug_bool
4760 {
4761   \iow_log:e
4762   {
4763     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4764     \l__luamplib_BBox_llx_tl\c_space_tl
4765     \l__luamplib_BBox_lly_tl\c_space_tl
4766     \l__luamplib_BBox_urx_tl\c_space_tl
4767     \l__luamplib_BBox_ury_tl
4768   }
4769   \sys_if_output_pdf:TF
4770   {
4771     \tl_set:Ne \l__luamplib_tag_bboxdraw_tl
4772     {

```

```

4773     \pdfextension save\relax
4774     \opacity_select:n{0.5} \color_select:n{red}
4775     \pdfextension literal~text
4776     {
4777         \l__luamplib_BBox_llx_tl\c_space_tl
4778         \l__luamplib_BBox_lly_tl\c_space_tl
4779         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4780         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4781         re~f
4782     }
4783     \pdfextension restore\relax
4784 }
4785 }
4786 {
4787     \tl_set:Nc \l__luamplib_tag_bbox_draw_tl
4788     {
4789         \special{pdf:bcontent}
4790         \opacity_select:n{0.5} \color_select:n{red}
4791         \special{pdf:code~
4792             1~0~0~1~
4793             -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4794             -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4795             cm
4796         }
4797         \special{pdf:code~
4798             \l__luamplib_BBox_llx_tl\c_space_tl
4799             \l__luamplib_BBox_lly_tl\c_space_tl
4800             \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4801             \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4802             re~f
4803         }
4804         \special{pdf:econtent}
4805     }
4806 }
4807 }
4808 }

```

Sockets for main process

```

4809 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4810 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4811 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4812 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4813 {
4814     \tag_mc_end_push:
4815     \tl_if_empty:NT\l__luamplib_tag_alt_tl
4816     {
4817         \tl_if_empty:eTF{#1}
4818         { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4819         { \tl_set:Nc \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }

```

```

4820     \msg_warning:nnVV{luamplib}{alt-text-missing}
4821         \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4822     }
4823     \tag_struct_begin:n
4824     {
4825         tag=\l__luamplib_tag_struct_tl,
4826         alt=\l__luamplib_tag_alt_tl,
4827     }
4828     \tag_mc_begin:n{}
4829 }
4830 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4831 {
4832     \cs_if_eq:NNT \luamplibexternalizemaybe \relax
4833     {
4834         \__luamplib_tag_bbox_attribute:n {#1}
4835     }
4836     #2
4837     \tl_use:N \l__luamplib_tag_bbox_draw_tl
4838     \tag_mc_end:
4839     \tag_struct_end:
4840     \tag_mc_begin_pop:n{}
4841 }
4842 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4843 {
4844     \tag_mc_end_push:
4845     \tag_struct_begin:n
4846     {
4847         tag=Span,
4848         actualtext=\l__luamplib_tag_actual_tl,
4849     }
4850     \tag_mc_begin:n{}
4851 }
4852 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4853 {
4854     #2
4855     \tag_mc_end:
4856     \tag_struct_end:
4857     \tag_mc_begin_pop:n{}
4858 }
4859 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4860 {
4861     \tag_mc_end_push:
4862     \tag_mc_begin:n{artifact}
4863 }
4864 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4865 {
4866     #2
4867     \tag_mc_end:
4868     \tag_mc_begin_pop:n{}

```

4869 }

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

```
4870 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4871 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4872 {
4873   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4874   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4875 }
4876 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4877 {
4878   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4879   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}
```

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

```
4880 \prependtomplibbox \mplibnoforcehmode
4881 \mode_if_vertical:T { \noindent \aftergroup\par }
4882 }
4883 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4884 {
4885   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4886   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4887 }
4888 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4889 {
4890   \bool_set_true:N \l__luamplib_tag_usetext_bool
4891   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4892   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4893   \prependtomplibbox \mplibnoforcehmode
4894   \mode_if_vertical:T { \noindent \aftergroup\par }
4895 }
4896 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4897 {
4898   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4899   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4900 }
4901 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
```

Key-value options

```
4902 \keys_define:nn{luamplib/tagging}
4903 {
4904   ,alt .code:n =
4905   {
4906     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4907     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4908   }
4909   ,actualtext .code:n =
4910   {
4911     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
```

```

4912 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4913 }
4914 ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4915 ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4916 ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4917 ,tag .code:n =
4918 {
4919 \str_case:nnF {#1}
4920 {
4921 {false} { \keys_set:nn {luamplib/tagging} {off} }
4922 {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4923 }
4924 {
4925 \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4926 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4927 }
4928 }
4929 ,adjust-BBox .code:n =
4930 {
4931 \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4932 \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4933 }
4934 ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4935 }
4936 \keys_define:nn {luamplib/instance}
4937 {
4938 ,externalize .code:n = { \cs_set_nopar:Npn \luamplibexternalizedothis {#1} }
4939 ,externalize .default:n = { true }
4940 ,externalizedependson .code:n = { \cs_set_nopar:Npn \luamplibexternalizedependson {#1} }
4941 ,externalizemargin .code:n = { \cs_set_nopar:Npn \luamplibexternalizemargin {#1} }
4942 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4943 ,instancename .meta:n = { instance = {#1} }
4944 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4945 }

```

Redefine our macros

```

4946 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4947 {
4948 \prependtomplibbox
4949 \hbox dir~TLT~\luamplibexternalizemaybe \bgroup
4950 \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4951 \xdef\MPllx{#1}\xdef\MPlly{#2}%
4952 \xdef\MPurx{#3}\xdef\MPury{#4}%
4953 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4954 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4955 \parskip0pt
4956 \leftskip0pt
4957 \parindent0pt
4958 \everypar{}%

```

```

4959 \setbox\mplibscratchbox\vbox\bgroup
4960 \tag_suspend:n{\mplibstarttoPDF}
4961 \noindent
4962 }
4963 \cs_set_nopar:Npn \mplibstoptoPDF
4964 {
4965 \par
4966 \egroup
4967 \setbox\mplibscratchbox\hbox
4968 {\hskip-\MPllx bp
4969 \raise-\MPlly bp
4970 \box\mplibscratchbox}%
4971 \setbox\mplibscratchbox\vbox to \MPheight
4972 {\vfill
4973 \hsize\MPwidth
4974 \wd\mplibscratchbox\z@
4975 \ht\mplibscratchbox\z@
4976 \dp\mplibscratchbox\z@
4977 \box\mplibscratchbox}%
4978 \wd\mplibscratchbox\MPwidth
4979 \ht\mplibscratchbox\MPheight
4980 \tag_socket_use:nnn{\luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4981 \egroup
4982 }
4983 \RenewDocumentCommand\mplibcode{0{}}
4984 {
4985 \tl_set:Nn \l__luamplib_tag_envname_tl {\mplibcode}
4986 \tl_gclear:N \currentmpinstancename
4987 \keys_set_known:neN {\luamplib/tagging} {#1} \l_tmpa_tl
4988 \keys_set:nV {\luamplib/instance} \l_tmpa_tl
4989 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4990 \tag_socket_use:n{\luamplib/figure/init}
4991 \mplibtmptoks{}\ltxdomplibcode
4992 }
4993 \RenewDocumentCommand\mpfig{s 0{}}
4994 {
4995 \group_begin:
4996 \tl_set:Nn \l__luamplib_tag_envname_tl {\mpfig}
4997 \keys_set_known:neN {\luamplib/tagging} {#2} \l_tmpa_tl
4998 \keys_set:nV {\luamplib/instance} \l_tmpa_tl
4999 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
5000 \tag_socket_use:n{\luamplib/figure/init}
5001 \IfBooleanTF{#1} { \mplibprempfig * }
5002 { \mplibmainmpfig }
5003 }
5004 \RenewDocumentCommand\usemplibgroup{0{ } m}
5005 {
5006 \group_begin:
5007 \tl_set:Nn \l__luamplib_tag_envname_tl {\usemplibgroup}

```

```

5008 \keys_set_known:nN {luamplib/tagging} {#1} \l_tmpa_tl
5009 \keys_set:nV {luamplib/instance} \l_tmpa_tl
5010 \tag_socket_use:n{luamplib/figure/init}
5011 \luamplib@externalized{luamplib.group.#2}
5012 \int_compare:nNnT {\l_tmpa_int} = {\c_zero_int}
5013 {
5014   \prependtomplibbox\hbox dir~TLT~\luamplibexternalizemaybe \bgroup
5015   \tag_socket_use:nn{luamplib/figure/begin}{#2}
5016   \setbox\mplibscratchbox\hbox\bgroup
5017   \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
5018   \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
5019   \egroup
5020   \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
5021   \egroup
5022 }
5023 \group_end:
5024 }

```

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

```

5025 \cs_new_nopar:Npn \mplibalttext #1
5026 {
5027   \tl_set:Ne \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
5028 }
5029 \cs_new_nopar:Npn \mplibactualtext #1
5030 {
5031   \tl_set:Ne \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
5032 }
5033 \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

1. 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.
2. 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.
3. 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:
- (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- (b) 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.
- (c) 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.

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

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

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

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

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

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

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

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

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.

11. 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

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

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.