

# The luamplib package

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

Current Maintainer: Kim Dohyun

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

2024/12/08 v2.36.0

## Abstract

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

## 1 Documentation

This package aims at providing a simple way to typeset directly METAPOST code in a document with LuaTeX. LuaTeX is built with the Lua mplib library, that runs METAPOST code. This package is basically a wrapper for the Lua mplib functions and some TeX 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  $\text{\LaTeX}$  in the `mplibcode` environment.

The resulting METAPOST figures are put in a TeX 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 ConTeXt. They have been adapted to  $\text{\LaTeX}$  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()` is a more versatile macro equivalent to `TEX()` 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`, though it's behavior cannot be the same as the stand-alone `mpost`. Of course 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](#).
- in the past, the package required PDF mode in order to have some output. Starting with version 2.7 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 T<sub>E</sub>X

**\mplibforcehmode** When this macro is declared, every METAPOST figure box will be typeset in horizontal mode, so `\centering`, `\raggedleft` etc will have effects. `\mplibnoforcehmode`, being default, reverts this setting. (Actually these commands redefine `\prependtomplibbox`; you can redefine this command with anything suitable before a box.)

**\everymplib{...}, \everyendmplib{...}** `\everymplib` and `\everyendmplib` redefine the lua table 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}
```

**\mplibsetformat{plain|metafun}** 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 functionalities producing 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](#)). 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=<number>"` to the sentence. ( $0 \leq \text{<number>} \leq 1$ )

From v2.36, `withtransparency(<number>|<string>,<number>)` is provided for *plain* format. The first argument accepts a number or a name of alternative transparency methods (see metafun manual Figure 8.1). The second argument accepts a number denoting opacity.

```
fill fullcircle scaled 10
  withcolor red
  withtransparency (1, 0.5)      % or ("normal", 0.5)
;
```

**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<sub>E</sub>X 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. The syntax is exactly the same as *metafun*'s new method (§ 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* (pictures made by `btex ... etex`, `texttext`, `maketext`, `infont`, etc) as well as paths can have shading effect.

```
draw btex \bfseries\TeX etex scaled 10
  withshadingmethod "linear"
  withshadingcolors (red,blue) ;
```

- When you give shading effect to a picture made by ‘`infont`’ operator, the result of `withshadingvector` will be the same as that of `withshadingdirection`, as `luamplib` considers only the bounding box of the picture.
- Inside tiling pattern cells (see [below § 1.2](#)), you shall not give shading effect to pictures (paths are OK). Anyway, that is the current phase of development.

**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 Reader, cannot properly render the isolated or knockout effect. Transparency group is available with *plain* format as well, with extended functionality. See [below § 1.2](#).

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

**\mplibshowlog{enable|disable}** Default: `disable`. When `\mplibshowlog{enable}`<sup>1</sup> is declared, log messages returned by the `METAPOST` process will be printed to the `.log` file. This is the  $\TeX$  side interface for `luamplib.showlog`.

**\mpliblegacybehavior{enable|disable}** 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.

```
\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 `verbatimtex ... etex`.

On the other hand,  $\TeX$  code in `verbatimtex ... etex` between `beginfig()` and `endfig` will be inserted after flushing out the `METAPOST` figure. As shown in the example below, `VerbatimTeX()` is a synonym of `verbatimtex ... etex`.

```
\mplibcode
```

---

<sup>1</sup>As for user's setting, `enable`, `true` and `yes` are identical; `disable`, `false` and `no` are identical.

```

D := sqrt(2)**7;
beginfig(0);
draw fullcircle scaled D;
VerbatimTeX("\gdef\Dia{" & decimal D & "}");
endfig;
\endmplibcode
diameter: \Dia bp.

```

By contrast, when `\mpliblegacybehavior{disable}` is declared, any `verbatimtex ... etex` will be executed, along with `btex ... etex`, sequentially one by one. So, some  $\TeX$  code in `verbatimtex ... etex` will have effects on following `btex ... etex` codes.

```

\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}

```

**`\mplibtexttextlabel{enable|disable}`** Default: `disable`. `\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)`.

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  $\TeX$ .

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

**Separate `METAPOST` instances** `luamplib` v2.22 has added the support for several named `METAPOST` instances in  $\mathbb{L}\TeX$  `mplibcode` environment. Plain  $\TeX$  users also can use this functionality. The syntax for  $\mathbb{L}\TeX$  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 environments with no instance name set (since if a name is set, the code is intended to be reused at some point).

- `btex ... etex` boxes are also shared and do not require `\mplibglobaltexttext`.
- When an instance names is set, respective `\currentmpinstancename` is set as well.

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

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

**`\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. This optional 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
```

**`\mplibverbatim{enable|disable}`** Default: `disable`. 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 [below](#)), all other `TEX` commands outside of the `btex` or `verbatimtex ... etex` are not expanded and will be fed literally to the `mplib` library.

**`\mpdim{...}`** 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.

```
\begin{mplibcode}
  beginfig(1)
  draw origin--(.6\mpdim{\linewidth},0) withpen pencircle scaled 4
  dashed evenly scaled 4 withcolor \mpcolor{orange};
endfig;
\end{mplibcode}
```

**`\mpcolor[...]{...}`** With `\mpcolor` command, color names or expressions of `color`, `xcolor` and `l3color` module/packages can be used in the `mplibcode` environment (after `withcolor` operator). See the example [above](#). The optional [...] denotes the option of `xcolor`'s `\color` command. For spot colors, `l3color` (in PDF/DVI mode), `colorspace`, `spotcolor` (in PDF mode) and `xespotcolor` (in DVI mode) packages are supported as well.

**\mpfig ... \endmpfig** Besides the `mplibcode` environment (for  $\TeX$ ) 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(withcolor .5[red,white]); }
\mpfig* input boxes \endmpfig
\mpfig
circleit.a(btex Box 1 etex); drawboxed(a);
\endmpfig
```

The instance name (default: `@mpfig`) can be changed 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{true}` is not declared.

**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  $\text{Lua}\TeX$ 's `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 excluding `.mp` extension. Note that `.mp` files under `$TEXMFMAIN/metapost/base` and `$TEXMFMAIN/metapost/context/base` are already registered by default.

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.

**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.

**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.

**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. Like the  $\TeX$ 's `picture` environment, available optional keys are `tag`, `alt`, `actualtext`, `artifact`, `debug` and `correct-BBox` (`texdoc latex-lab-graphic`). Additionally, `luamplib` provides its own `text` key.

`tag=...` You can choose a tag name, default value being `Figure`. BBox info will be added automatically to the PDF unless the value is `text` or `false`. When the value is `false`, tagging is deactivated.

`debug` draws bounding box of the figure for checking, which you can correct by `correct-BBox` key with space-separated four dimen values.

`alt=...` sets an alternative text of the figure as given. This key is needed for ordinary `METAPOST` figures. You can give alternative text within `METAPOST` code as well: `VerbatimTeX{"\mplibalttext{...}"};`

`actualtext=...` starts a `Span` tag implicitly and sets an actual text as given. Horizontal mode is forced by `\noindent` command. 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 give actual text within `METAPOST` code as well: `VerbatimTeX{"\mplibactualtext{...}"};`

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

`text` starts an artifact MC and enables tagging on `texttext` (the same as `btex ... etex`) boxes. Horizontal mode is forced by `\noindent` command. BBox info will not be added. This key is intended for figures made mostly of `texttext` boxes. Inside text-keyed figures, reusing `texttext` boxes is strongly discouraged.

These keys are provided also for `\mpfig` and `\usemplibgroup` (see [below](#)) commands.

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

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

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

```

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

```

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

## 1.2 METAPOST

**mplibdimen(...)**, **mplibcolor(...)** These are METAPOST interfaces for the  $\TeX$  commands `\mpdim` and `\mpcolor` (see [above](#)). 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  $\TeX$  commands outside of the `btex` or `verbatimtex` ... `etex`.

**mplibtexcolor ...**, **mplibrgbtexcolor ...** `mplibtexcolor`, which accepts a string argument, is a METAPOST operator that converts a  $\TeX$  color expression to a METAPOST color expression, that can be used anywhere color expression is expected as well as after the `withcolor` operator. For instance:

```

color col;
col := mpibtexcolor "olive!50";

```

But the result may vary in its color model (gray/rgb/cmyk) according to the given  $\TeX$  color. (Spot colors are forced to cmyk model, so this operator is not recommended for spot colors.) 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.

**mplibgraphicstext ...** `mplibgraphicstext` is a METAPOST operator, the effect of which is similar to that of  $\text{Con}\TeX$ t's `graphicstext` or our own `mpliboutlinetext` (see [below](#)). However the syntax is somewhat different.

```

mpibgraphicstext "Funny"
fakebold 2.3          % fontspec option
drawcolor .7blue fillcolor "red!50" % color expressions

```

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

N.B. In some cases, `mplibgraphicstext` will produce better results than  $\text{Con}\TeX$ t or even than our own `mpliboutlinetext`, especially when processing complicated  $\TeX$  code such as the vertical writing in Chinese or Japanese. However, because the implementation is quite different from others, there are some limitations such that you can't apply shading (gradient colors) to the text. Again, in DVI mode, `unicode-math` package is needed for math formula, as we cannot embolden type1 fonts in DVI mode.



**mplibglyph ... of ...** From v2.30, we provide a new METAPOST operator `mplibglyph`, which returns a METAPOST picture containing outline paths of a glyph in opentype, true-type or type1 fonts. When a type1 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 "Q" of "Times.ttc(2)"           % subfont number
mplibglyph "Q" of "SourceHanSansK-VF.otf[Regular]" % instance name
```

Both arguments before and after of “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<sub>E</sub>X font csname (without backslash) or the raw filename of a font. When it is a font filename, a number within parentheses after the filename denotes a subfont number (starting from zero) of a TTC font; a string within brackets denotes an instance name of a variable font.

**mplibdrawglyph ...** The picture returned by `mplibglyph` will be quite similar to the result of glyph primitive in its structure. So, 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.

☞ 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, with *plain* format as well, additionally declare `withpostscript "evenodd"` to the last path in the picture.

**mpliboutlinetext (...)** From v2.31, a new METAPOST operator `mpliboutlinetext` is available, which mimicks *metafun*’s `outlinetext`. So the syntax is the same: see the *metafun* manual § 8.7 (texdoc *metafun*). A simple example:

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

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 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.

**\mppattern{...} ... \endmppattern, ... withpattern ...** T<sub>E</sub>X macros `\mppattern{<name>}` ... `\endmppattern` define a tiling pattern associated with the <name>. METAPOST operator `withpattern`, the syntax being `<path>|<textual picture> withpattern <string>`, will return a METAPOST picture which fills the given path or text with a tiling pattern of the <name> by replicating it horizontally and vertically. The *textual picture* here means any text typeset by T<sub>E</sub>X, mostly the result of the `btex` command (though technically this is not a true textual picture) or the `infont` operator.

Table 1: options for \mppattern

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

\* in string type, numbers are separated by spaces

An example:

```

\mppattern{mypatt}           % or \begin{mppattern}{mypatt}
[                             % options: see below
  xstep = 10,
  ystep = 12,
  matrix = {0, 1, -1, 0},    % or "0 1 -1 0"
]
\mpfig                       % or any other TeX code,
  draw (origin--(1,1))
    scaled 10
    withcolor 1/3[blue,white]
  ;
  draw (up--right)
    scaled 10
    withcolor 1/3[red,white]
  ;
\endmpfig
\endmppattern                % or \end{mppattern}

\mpfig
  draw fullcircle scaled 90
    withpostscript "collect"
  ;
  draw fullcircle scaled 200
    withpattern "mypatt"
    withpen pencircle scaled 1
    withcolor \mpcolor{red!50!blue!50}
    withpostscript "evenodd"
  ;
\endmpfig

```

The available options are listed in Table 1.

For the sake of convenience, the width and height values of tiling patterns 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 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 effects such as transparency in a pattern, resources option is needed: for instance, resources="/ExtGState 1 0 R". However, as luamplib automatically includes the resources of the current page, this option is not needed in most cases.

Option colored=false (coloured is a synonym of colored) will generate an uncolored pattern which shall have no color at all. Uncolored pattern will be painted later by the color of a METAPOST object. An example:

```
\begin{mppattern}{pattnocolor}
[
  colored = false,
  matrix = "slanted .3 rotated 30",
]
\tiny\TeX
\end{mppattern}

\begin{mplibcode}
beginfig(1)
picture tex;
tex = mpliboutlinetext.p ("bfseries \TeX");
for i=1 upto mpliboutlinenum:
  j:=0;
  for item within mpliboutlinepic[i]:
    j:=j+1;
    draw pathpart item scaled 10
    if j < length mpliboutlinepic[i]:
      withpostscript "collect"
    else:
      withpattern "pattnocolor"
      withpen pencircle scaled 1/2
      withcolor (i/4)[red,blue]      % paints the pattern
    fi;
  endfor
endfor
endfig;
\end{mplibcode}
```

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

```
\begin{mplibcode}
beginfig(2)
picture pic;
pic = mplibgraphictext "\bfseries\TeX"
      fakebold 1/2
      fillcolor 1/3[red,blue]      % paints the pattern
      drawcolor 2/3[red,blue]
      scaled 10 ;
draw pic withpattern "pattnocolor" ;
endfig;
\end{mplibcode}
```

... **withfademethod** ... This is a METAPOST operator which makes the color of an object gradiently transparent. The syntax is  $\langle path \rangle | \langle picture \rangle$  withfademethod  $\langle string \rangle$ , the latter being either "linear" or "circular". Though it is similar to the withshademethod from

*metafun*, the differences are: (1) the operand of `withfademethod` can be a picture as well as a path; (2) you cannot make gradient colors, but can only make gradient opacity.

Related macros to control optional values are:

`withfadeopacity` (*number, number*) 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` (*number, number*) 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.

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

An example:

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

... **asgroup** ... As said [before](#), transparency group is available with *plain* as well as *metafun* format. The syntax is exactly the same: `<picture> | <path> asgroup "" | "isolated" | "knockout" | "isolated,knockout"`, which will return a METAPOST picture. 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 additional feature provided by `luamplib` is that you can reuse the group as many times as you want in the T<sub>E</sub>X code or in other METAPOST code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide T<sub>E</sub>X and METAPOST macros as follows:

`withgroupname` `<string>` associates a transparency group with the given name. When this is not appended to the sentence with `asgroup` operator, the default group name ‘`lastmplibgroup`’ will be used.

`\usemplibgroup{...}` is a T<sub>E</sub>X command to reuse a transparency group of the name once used. Note that the position of the group will be origin-based: in other words, lower-left corner of the group will be shifted to the origin.

`usemplibgroup <string>` is a METAPOST command which will add a transparency group of the name to the currentpicture. Contrary to the T<sub>E</sub>X command just mentioned, the position of the group is the same as the original transparency group.

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

An example showing the difference between the T<sub>E</sub>X and METAPOST commands:

```
\mpfig
draw image(
  fill fullcircle scaled 100 shifted 25right withcolor blue;
  fill fullcircle scaled 100 withcolor red ;
) asgroup ""
  withgroupname "mygroup";
draw (left--right) scaled 10;
draw (up--down) scaled 10;
\endmpfig

\noindent
\clap{\vrule width 20pt height .25pt depth .25pt}%
\clap{\vrule width .5pt height 10pt depth 10pt}%
\usemplibgroup{mygroup}

\mpfig
usemplibgroup "mygroup" rotated 15
  withtransparency (1, 0.5) ;
draw (left--right) scaled 10;
draw (up--down) scaled 10;
\endmpfig
```

Also note that normally the reused transparency groups are not affected by outer color commands. However, if you have made the original transparency group using `withoutcolor` command, colors will have effects on the uncolored objects in the group.

**`\mplibgroup{...}` ... `\endmplibgroup`** These T<sub>E</sub>X macros are described here in this subsection, as they are deeply related to the `asgroup` operator. Users can define a transparency group or a normal *form XObject* with these macros from T<sub>E</sub>X side. The syntax is similar to the `\mppattern` command (see [above](#)). An example:

```
\mplibgroup{mygrx}           % or \begin{mplibgroup}{mygrx}
[                             % options: see below
  asgroup="",
]
\mpfig                       % or any other TeX code
pickup pencircle scaled 10;
draw (left--right) scaled 30 rotated 45 ;
```

Table 2: options for `\mplibgroup`

Key	Value Type	Explanation
<code>asgroup</code>	<i>string</i>	<code>""</code> , <code>"isolated"</code> , <code>"knockout"</code> , or <code>"isolated, knockout"</code>
<code>bbox</code>	<i>table</i> or <i>string</i>	<code>llx</code> , <code>lly</code> , <code>urx</code> , <code>ury</code> values *
<code>matrix</code>	<i>table</i> or <i>string</i>	<code>xx</code> , <code>yx</code> , <code>xy</code> , <code>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

```

    draw (left--right) scaled 30 rotated -45 ;
\endmpfig
\endmplibgroup           % or \end{mplibgroup}

\usemplibgroup{mygrx}

\mpfig
  usemplibgroup "mygrx" scaled 1.5
  withtransparency (1, 0.5) ;
\endmpfig

```

Available options, much fewer than those for `\mppattern`, are listed in Table 2. Again, the width/height/depth values of the `mplibgroup` will be written down into the log file.

When `asgroup` option, including empty string, is not given, a normal 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.

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

### 1.3 Lua

**runscript ...** Using the primitive `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` value 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.

**Lua table `luamplib.instances`** 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 Lua $\TeX$  manual § 11.2.8.4 (texdoc `luatex`). The following will print `false`, `3.0`, `MetaPost` and the knots and the cyclicity of the path `unitsquare`, consecutively.

```

\begin{mplibcode}[instance1]
  boolean b; b = 1 > 2;
  numeric n; n = 3;

```

Table 3: elements in luamplib table (partial)

Key	Type	Related T <sub>E</sub> X macro
codeinherit	<i>boolean</i>	<code>\mplibcodeinherit</code>
everyendmplib	<i>table</i>	<code>\everyendmplib</code>
everymplib	<i>table</i>	<code>\everymplib</code>
getcachedir	<i>function</i> (<string>)	<code>\mplibcachedir</code>
globaltexttext	<i>boolean</i>	<code>\mplibglobaltexttext</code>
legacyverbatimtex	<i>boolean</i>	<code>\mpliblegacybehavior</code>
noneedtoreplace	<i>table</i>	<code>\mplibmakenocache</code>
numbersystem	<i>string</i>	<code>\mplibnumbersystem</code>
setformat	<i>function</i> (<string>)	<code>\mplibsetformat</code>
showlog	<i>boolean</i>	<code>\mplibshowlog</code>
texttextlabel	<i>boolean</i>	<code>\mplibtexttextlabel</code>
verbatiminput	<i>boolean</i>	<code>\mplibverbatim</code>

```

string s; s = "MetaPost";
path p; p = unitsquare;
\end{mplibcode}

\directlua{
  local instance1 = luamplib.instances.instance1
  print( instance1:get_boolean "b" )
  print( instance1:get_number  "n" )
  print( instance1:get_string  "s" )
  local t = instance1:get_path "p"
  for k,v in pairs(t) do
    print(k, type(v)=='table' and table.concat(v, ' ') or v)
  end
}

```

**Lua function `luamplib.process_mplibcode`** Users can execute 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 have effects on the process of `process_mplibcode`.

## 2 Implementation

### 2.1 Lua module

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.36.0",
5   date      = "2024/12/08",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",

```

```

7 }
8

```

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

```

9 luamplib      = luamplib or { }
10 local luamplib = luamplib
11
12 local format, abs = string.format, math.abs
13
14 Use our own function for warn/info/err.
15 local function termorlog (target, text, kind)
16   if text then
17     local mod, write, append = "luamplib", texio.write_nl, texio.write
18     kind = kind
19     or target == "term" and "Warning (more info in the log)"
20     or target == "log" and "Info"
21     or target == "term and log" and "Warning"
22     or "Error"
23     target = kind == "Error" and "term and log" or target
24     local t = text:explode"\n+"
25     write(target, format("Module %s %s:", mod, kind))
26     if #t == 1 then
27       append(target, format(" %s", t[1]))
28     else
29       for _,line in ipairs(t) do
30         write(target, line)
31       end
32       write(target, format("(%s)", mod))
33     end
34     append(target, format(" on input line %s", tex.inputlineno))
35     write(target, "")
36     if kind == "Error" then error() end
37   end
38 end
39 local function warn (...) -- beware '%' symbol
40   termorlog("term and log", select("#",...) > 1 and format(...) or ...)
41 end
42 local function info (...)
43   termorlog("log", select("#",...) > 1 and format(...) or ...)
44 end
45 local function err (...)
46   termorlog("error", select("#",...) > 1 and format(...) or ...)
47 end
48 luamplib.showlog = luamplib.showlog or false
49

```

This module is a stripped down version of libraries that are used by `ConTeXt`. 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 teksprint   = tex.sprint
54 local texgettoks  = tex.gettoks

```



```

55 local texgetbox = tex.getbox
56 local texruntoks = tex.runtoks
57 if not texruntoks then
58   err("Your LuaTeX version is too old. Please upgrade it to the latest")
59 end
60 local is_defined = token.is_defined
61 local get_macro = token.get_macro
62 local mplib = require ('mplib')
63 local kpse = require ('kpse')
64 local lfs = require ('lfs')
65 local lfsattributes = lfs.attributes
66 local lfsisdir = lfs.isdir
67 local lfsmkdir = lfs.mkdir
68 local lfstouch = lfs.touch
69 local ioopen = io.open
70

```

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

```

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

```

93 local luamplibtime = lfsattributes(kpse.find_file"luamplib.lua", "modification")
94 local currenttime = os.time()
95 local outputdir, cachedir
96 if lfstouch then
97   for i,v in ipairs{'TEXMFVAR', 'TEXMF_OUTPUT_DIRECTORY', '.', 'TEXMFOUTPUT'} do
98     local var = i == 3 and v or kpse.var_value(v)
99     if var and var ~= "" then
100       for _,vv in next, var:explode(os.type == "unix" and ":" or ";") do
101         local dir = format("%s/%s", vv, "luamplib_cache")
102         if not lfsisdir(dir) then
103           mk_full_path(dir)
104         end

```

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

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

    format.mp is much complicated, so specially treated.
148 local function replaceformatmp(file,newfile,ofmodify)
149     local fh = ioopen(file,"r")
150     if not fh then return file end
151     local data = fh:read("*all"); fh:close()
152     fh = ioopen(newfile,"w")
153     if not fh then return file end
154     fh:write(
155         "let normalinfont = infont;\n",

```

```

156     "primarydef str infont name = rawtexttext(str) enddef;\n",
157     data,
158     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
159     "vardef Fexp_(expr x) = rawtexttext(\"$^{\"&decimal x&\"}$\") enddef;\n",
160     "let infont = normalinfont;\n"
161 ); fh:close()
162 lfstouch(newfile,currenttime,ofmodify)
163 return newfile
164 end

    Replace btex ... etex and verbatimtex ... etex in input files, if needed.
165 local name_b = "%f[%a_]"
166 local name_e = "%f[^%a_]"
167 local btex_etex = name_b.."btex"..name_e.."s*(.)s*"..name_b.."etex"..name_e
168 local verbatimtex_etex = name_b.."verbatimtex"..name_e.."s*(.)s*"..name_b.."etex"..name_e
169 local function replaceinputmpfile (name,file)
170     local ofmodify = lfsattributes(file,"modification")
171     if not ofmodify then return file end
172     local newfile = name:gsub("%W","_")
173     newfile = format("%s/luamplib_input%s", cachedir or outputdir, newfile)
174     if newfile and luamplibtime then
175         local nf = lfsattributes(newfile)
176         if nf and nf.mode == "file" and
177             ofmodify == nf.modification and luamplibtime < nf.access then
178             return nf.size == 0 and file or newfile
179         end
180     end
181     if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
182     local fh = ioopen(file,"r")
183     if not fh then return file end
184     local data = fh:read("*all"); fh:close()

    "etex" must be preceded by a space and followed by a space or semicolon as specified in
    LuaTeX manual, which is not the case of standalone METAPOST though.
185     local count,cnt = 0,0
186     data, cnt = data:gsub(btex_etex, "btex %1 etex ") -- space
187     count = count + cnt
188     data, cnt = data:gsub(verbatimtex_etex, "verbatimtex %1 etex;") -- semicolon
189     count = count + cnt
190     if count == 0 then
191         needtoreplace[name] = true
192         fh = ioopen(newfile,"w");
193         if fh then
194             fh:close()
195             lfstouch(newfile,currenttime,ofmodify)
196         end
197         return file
198     end
199     fh = ioopen(newfile,"w")
200     if not fh then return file end
201     fh:write(data); fh:close()
202     lfstouch(newfile,currenttime,ofmodify)
203     return newfile
204 end
205

```

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.

```

206 local mpkpse
207 do
208   local exe = 0
209   while arg[exe-1] do
210     exe = exe-1
211   end
212   mpkpse = kpse.new(arg[exe], "mpost")
213 end
214 local special_ftype = {
215   pfb = "type1 fonts",
216   enc = "enc files",
217 }
218 function luamplib.finder (name, mode, ftype)
219   if mode == "w" then
220     if name and name ~= "mpout.log" then
221       kpse.record_output_file(name) -- recorder
222     end
223     return name
224   else
225     ftype = special_ftype[ftype] or ftype
226     local file = mpkpse.find_file(name,ftype)
227     if file then
228       if lfstouch and ftype == "mp" and not noneedtoreplace[name] then
229         file = replaceinputmpfile(name,file)
230       end
231     else
232       file = mpkpse.find_file(name, name:match("%a+$"))
233     end
234     if file then
235       kpse.record_input_file(file) -- recorder
236     end
237     return file
238   end
239 end
240
```

Create and load mplib instances. We do not support ancient version of mplib any more. (Don't know which version of mplib started to support make\_text and run\_script; let the users find it.)

```

241 local preamble = [[
242   boolean mplib ; mplib := true ;
243   let dump = endinput ;
244   let normalfontsize = fontsize;
245   input %s ;
246 ]]

```

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

```

247 local currentformat = "plain"
248 function luamplib.setformat (name)
249   currentformat = name
250 end

```

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

```

251 luamplib.codeinherit = false

```

```

252 local mplibinstances = {}
253 luamplib.instances = mplibinstances
254 local has_instancename = false
255 local function reporterror (result, prevlog)
256   if not result then
257     err("no result object returned")
258   else
259     local t, e, l = result.term, result.error, result.log
log has more information than term, so log first (2021/08/02)
260     local log = l or t or "no-term"
261     log = log:gsub("%(Please type a command or say `end'%)", ""):gsub("\n+", "\n")
262     if result.status > 0 then
263       local first = log:match"(-\n! .-)\n! "
264       if first then
265         termorlog("term", first)
266         termorlog("log", log, "Warning")
267       else
268         warn(log)
269       end
270       if result.status > 1 then
271         err(e or "see above messages")
272       end
273     elseif prevlog then
274       log = prevlog..log

```

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

```

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

```

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

```

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

```

Make use of make\_text and run\_script, which will co-operate with LuaTeX's tex.runtoks or other Lua functions. And we provide numbersystem option since v2.4. See <https://github.com/lualatex/luamplib/issues/21>.

```

291   make_text   = luamplib.maketext,
292   run_script  = luamplib.runscript,
293   math_mode   = luamplib.numbersystem,
294   job_name    = tex.jobname,
295   random_seed = math.random(4095),

```

```

296     extensions = 1,
297 }

```

Append our own METAPOST preamble to the preamble above.

```

298 local preamble = tableconcat{
299     format(preamble, replacesuffix(name,"mp")),
300     luamplib.preambles.mplibcode,
301     luamplib.legacyverbatimtex and luamplib.preambles.legacyverbatimtex or "",
302     luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
303 }
304 local result, log
305 if not mpx then
306     result = { status = 99, error = "out of memory"}
307 else
308     result = mpx:execute(preamble)
309 end
310 log = reporterror(result)
311 return mpx, result, log
312 end

```

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

```

313 local function process (data, instancename)
314     local currfmt
315     if instancename and instancename ~= "" then
316         currfmt = instancename
317         has_instancename = true
318     else
319         currfmt = tableconcat{
320             currentformat,
321             luamplib.numbersystem or "scaled",
322             tostring(luamplib.texttextlabel),
323             tostring(luamplib.legacyverbatimtex),
324         }
325         has_instancename = false
326     end
327     local mpx = mplibinstances[currfmt]
328     local standalone = not (has_instancename or luamplib.codeinherit)
329     if mpx and standalone then
330         mpx:finish()
331     end
332     local log = ""
333     if standalone or not mpx then
334         mpx, _, log = luamplibload(currentformat)
335         mplibinstances[currfmt] = mpx
336     end
337     local converted, result = false, {}
338     if mpx and data then
339         result = mpx:execute(data)
340         local log = reporterror(result, log)
341         if log then
342             if result.fig then
343                 converted = luamplib.convert(result)
344             end
345         end
346     else

```

```

347     err"Mem file unloadable. Maybe generated with a different version of mplib?"
348 end
349 return converted, result
350 end
351

```

dvipdfmx is supported, though nobody seems to use it.

```

352 local pdfmode = tex.outputmode > 0
353

```

make\_text and some run\_script uses Lua<sub>TeX</sub>'s tex.runtoks.

```

354 local catlatex = luatexbase.registernumber("catcodetable@latex")
355 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.

```

356 local function run_tex_code (str, cat)
357     texruntoks(function() texsprint(cat or catlatex, str) end)
358 end

```

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.

```

359 local texboxes = { globalid = 0, localid = 4096 }

```

For conversion of sp to bp.

```

360 local factor = 65536*(7227/7200)
361 local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
362     xscaled %f yscaled %f shifted (0,-%f) \z
363     withprescript "mplibtexboxid=%i:%f:%f")'
364 local function process_tex_text (str, maketext)
365     if str then
366         if not maketext then str = str:gsub("\r.-$", "") end
367         local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
368             and "\global" or ""
369         local tex_box_id
370         if global == "" then
371             tex_box_id = texboxes.localid + 1
372             texboxes.localid = tex_box_id
373         else
374             local boxid = texboxes.globalid + 1
375             texboxes.globalid = boxid
376             run_tex_code(format([[\\expandafter\\newbox\\csname luamplib.box.%s\\endcsname]], boxid))
377             tex_box_id = tex.getcount'allocationnumber'
378         end
379         run_tex_code(format("\\luamplibtagtextbegin{%i}%s\\setbox%i\\hbox{%s}\\luamplibtagtextend", tex_box_id, global,
380             local box = texgetbox(tex_box_id)
381             local wd = box.width / factor
382             local ht = box.height / factor
383             local dp = box.depth / factor
384             return texttext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
385         end
386     return ""
387 end

```

388

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.

```

389 local mplibcolorfmt = {
390   xcolor = tableconcat{
391     [[\begingroup\let\XC@color\relax]],
392     [[\def\set@color{\global\mplibtmptoks\expandafter{\current@color}}]],
393     [[\color%s\endgroup]],
394   },
395   l3color = tableconcat{
396     [[\begingroup\def\__color_select:N#1{\expandafter\__color_select:nn#1}]],
397     [[\def\__color_backend_select:nn#1#2{\global\mplibtmptoks{#1 #2}}]],
398     [[\def\__kernel_backend_literal:e#1{\global\mplibtmptoks\expandafter{\expanded{#1}}}],
399     [[\color_select:n%s\endgroup]],
400   },
401 }
402 local colfmt = is_defined'color_select:n' and "l3color" or "xcolor"
403 if colfmt == "l3color" then
404   run_tex_code{
405     "\\newcatcodetable\\luamplibcctabexplat",
406     "\\begingroup",
407     "\\catcode`@=11 ",
408     "\\catcode`_=11 ",
409     "\\catcode`:=11 ",
410     "\\savecatcodetable\\luamplibcctabexplat",
411     "\\endgroup",
412   }
413 end
414 local ccexplat = luatexbase.registernumber"luamplibcctabexplat"
415 local function process_color (str)
416   if str then
417     if not str:find("%b{") then
418       str = format("{%s}",str)
419     end
420     local myfmt = mplibcolorfmt[colfmt]
421     if colfmt == "l3color" and is_defined"color" then
422       if str:find("%b[") then
423         myfmt = mplibcolorfmt.xcolor
424       else
425         for _,v in ipairs(str:match"{(.+)}:explode!") do
426           if not v:find("^%s*d+%s*$") then
427             local pp = get_macro(format("l__color_named_%s_prop",v))
428             if not pp or pp == "" then
429               myfmt = mplibcolorfmt.xcolor
430             break
431           end
432         end
433       end
434     end
435     run_tex_code(myfmt:format(str), ccexplat or catat11)
436     local t = texgettoks"mplibtmptoks"
437     if not pdfmode and not t:find"^pdf" then
438       t = t:gsub("%a+ (.+)", "pdf:bc [%1]")

```



```

440     end
441     return format('1 withprescript "mpliboverridecolor=%s"', t)
442 end
443 return ""
444 end
445
    for \mpdim or mplibdimen
446 local function process_dimen (str)
447   if str then
448     str = str:gsub("{(.+)}", "%1")
449     run_tex_code(format([[ \mplibtmp toks \expandafter { \the \dimexpr %s \relax } ]], str))
450     return format("begin group %s end group", texgettoks "mplibtmp toks")
451   end
452   return ""
453 end
454
    Newly introduced method of processing verbatimex ... etex. This function is used
    when \mpliblegacybehavior{false} is declared.
455 local function process_verbatimex_text (str)
456   if str then
457     run_tex_code(str)
458   end
459   return ""
460 end
461
    For legacy verbatimex process. verbatimex ... etex before beginfig() is not ig-
    nored, but the TeX code is inserted just before the mplib box. And TeX code inside
    beginfig() ... endfig is inserted after the mplib box.
462 local tex_code_pre_mplib = {}
463 luamplib.figid = 1
464 luamplib.in_the_fig = false
465 local function process_verbatimex_prefig (str)
466   if str then
467     tex_code_pre_mplib[luamplib.figid] = str
468   end
469   return ""
470 end
471 local function process_verbatimex_infig (str)
472   if str then
473     return format('special "postmplibverbtex=%s";', str)
474   end
475   return ""
476 end
477
478 local runscript_funcs = {
479   luamplibtext    = process_tex_text,
480   luamplibcolor   = process_color,
481   luamplibdimen   = process_dimen,
482   luamplibprefig  = process_verbatimex_prefig,
483   luamplibinfig   = process_verbatimex_infig,
484   luamplibverbtex = process_verbatimex_text,
485 }
486

```

For *metafun* format. see issue #79.

```
487 mp = mp or {}
488 local mp = mp
489 mp.mf_path_reset = mp.mf_path_reset or function() end
490 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
491 mp.report = mp.report or info
```

*metafun* 2021-03-09 changes crashes luamplib.

```
492 catcodes = catcodes or {}
493 local catcodes = catcodes
494 catcodes.numbers = catcodes.numbers or {}
495 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
496 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
497 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
498 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
499 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
500 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
501 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
502
```

A function from ConT<sub>E</sub>Xt general.

```
503 local function mpprint(buffer,...)
504   for i=1,select("#",...) do
505     local value = select(i,...)
506     if value ~= nil then
507       local t = type(value)
508       if t == "number" then
509         buffer[#buffer+1] = format("%.16f",value)
510       elseif t == "string" then
511         buffer[#buffer+1] = value
512       elseif t == "table" then
513         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
514       else -- boolean or whatever
515         buffer[#buffer+1] = tostring(value)
516       end
517     end
518   end
519 end
520 function luamplib.runscript (code)
521   local id, str = code:match("(.-){(.*)}")
522   if id and str then
523     local f = runscript_funcs[id]
524     if f then
525       local t = f(str)
526       if t then return t end
527     end
528   end
529   local f = loadstring(code)
530   if type(f) == "function" then
531     local buffer = {}
532     function mp.print(...)
533       mpprint(buffer,...)
534     end
535     local res = {f()}
536     buffer = tableconcat(buffer)
```

```

537   if buffer and buffer ~= "" then
538       return buffer
539   end
540   buffer = {}
541   mpprint(buffer, tableunpack(res))
542   return tableconcat(buffer)
543 end
544 return ""
545 end
546
    make_text must be one liner, so comment sign is not allowed.
547 local function protecttexcontents (str)
548   return str:gsub("\\%", "\\0PerCent\0")
549         :gsub("%%.\n", "")
550         :gsub("%%.-$", "")
551         :gsub("%zPerCent%z", "\\%")
552         :gsub("\\r.-$", "")
553         :gsub("%s+", " ")
554 end
555 luamplib.legacyverbatimex = true
556 function luamplib.maketext (str, what)
557   if str and str ~= "" then
558     str = protecttexcontents(str)
559     if what == 1 then
560       if not str:find("\\documentclass"..name_e) and
561         not str:find("\\begin%s*{document}") and
562         not str:find("\\documentstyle"..name_e) and
563         not str:find("\\usepackage"..name_e) then
564         if luamplib.legacyverbatimex then
565           if luamplib.in_the_fig then
566             return process_verbatimex_infig(str)
567           else
568             return process_verbatimex_prefig(str)
569           end
570         else
571           return process_verbatimex_text(str)
572         end
573       end
574     else
575       return process_tex_text(str, true) -- bool is for 'char13'
576     end
577   end
578   return ""
579 end
580
    luamplib's METAPOST color operators
581 local function colorsplit (res)
582   local t, tt = { }, res:gsub("[%[%]]", "", 2):explode()
583   local be = tt[1]:find"%d" and 1 or 2
584   for i=be, #tt do
585     if not tonumber(tt[i]) then break end
586     t[#t+1] = tt[i]
587   end

```

```

588 return t
589 end
590
591 luamplib.gettexcolor = function (str, rgb)
592   local res = process_color(str):match'"mpliboverridecolor=(.)"'
593   if res:find" cs " or res:find"@pdf.obj" then
594     if not rgb then
595       warn("%s is a spot color. Forced to CMYK", str)
596     end
597     run_tex_code({
598       "\\color_export:nnN{",
599       str,
600       "}{",
601       rgb and "space-sep-rgb" or "space-sep-cmyk",
602       "}"\\mplib@tempa",
603     },ccexplat)
604     return get_macro"mplib@tempa":explode()
605   end
606   local t = colorsplit(res)
607   if #t == 3 or not rgb then return t end
608   if #t == 4 then
609     return { 1 - math.min(1,t[1]+t[4]), 1 - math.min(1,t[2]+t[4]), 1 - math.min(1,t[3]+t[4]) }
610   end
611   return { t[1], t[1], t[1] }
612 end
613
614 luamplib.shadecolor = function (str)
615   local res = process_color(str):match'"mpliboverridecolor=(.)"'
616   if res:find" cs " or res:find"@pdf.obj" then -- spot color shade: 13 only

```

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}
\color_model_new:nnn { pantone2040 }
{ Separation }
{

```

```

        name = PANTONE~2040~U ,
        alternative-model = cmyk ,
        alternative-values = {0, 0.28, 0.21, 0.04}
    }
    \color_set:nnn{spotD}{pantone2040}{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","spotD")
        )
    ;
endfig;
\end{mplibcode}
\end{document}

```

another one: user-defined DeviceN colorspace

```

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

```

617    run\_tex\_code({

618        [[\color\_export:nnN{]], str, [[{backend}\mplib@tempa]],

```

619 },ccexplat)
620 local name, value = get_macro'mplib@tempa':match'{{(.-)}}{{(.-)}}'
621 local t, obj = res:explode()
622 if pdfmode then
623     obj = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
624 else
625     obj = t[2]
626 end
627 return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
628 end
629 return colorsplit(res)
630 end
631

```

Remove trailing zeros for smaller PDF

```

632 local decimals = "%.0d+"
633 local function rmzeros(str) return str:gsub("%.?0+$","") end
634

```

luamplib's mplibgraphicstext operator

```

635 local emboldenfonts = { }
636 local function getemboldenwidth (curr, fakebold)
637     local width = emboldenfonts.width
638     if not width then
639         local f
640         local function getglyph(n)
641             while n do
642                 if n.head then
643                     getglyph(n.head)
644                 elseif n.font and n.font > 0 then
645                     f = n.font; break
646                 end
647                 n = node.getnext(n)
648             end
649         end
650         getglyph(curr)
651         width = font.getcopy(f or font.current()).size * fakebold / factor * 10
652         emboldenfonts.width = width
653     end
654     return width
655 end
656 local function getrulewhatsit (line, wd, ht, dp)
657     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
658     local pl
659     local fmt = "%f w %f %f %f %f re %s"
660     if pdfmode then
661         pl = node.new("whatsit", "pdf_literal")
662         pl.mode = 0
663     else
664         fmt = "pdf:content " .. fmt
665         pl = node.new("whatsit", "special")
666     end
667     pl.data = fmt:format(line, 0, -dp, wd, ht+dp, "B") :gsub(decimals,rmzeros)
668     local ss = node.new"glue"
669     node.setglue(ss, 0, 65536, 65536, 2, 2)

```

```

670 pl.next = ss
671 return pl
672 end
673 local function getrulemetric (box, curr, bp)
674   local running = -1073741824
675   local wd,ht,dp = curr.width, curr.height, curr.depth
676   wd = wd == running and box.width or wd
677   ht = ht == running and box.height or ht
678   dp = dp == running and box.depth or dp
679   if bp then
680     return wd/factor, ht/factor, dp/factor
681   end
682   return wd, ht, dp
683 end
684 local function embolden (box, curr, fakebold)
685   local head = curr
686   while curr do
687     if curr.head then
688       curr.head = embolden(curr, curr.head, fakebold)
689     elseif curr.replace then
690       curr.replace = embolden(box, curr.replace, fakebold)
691     elseif curr.leader then
692       if curr.leader.head then
693         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
694       elseif curr.leader.id == node.id"rule" then
695         local glue = node.effective_glue(curr, box)
696         local line = getemboldenwidth(curr, fakebold)
697         local wd,ht,dp = getrulemetric(box, curr.leader)
698         if box.id == node.id"hlist" then
699           wd = glue
700         else
701           ht, dp = 0, glue
702         end
703         local pl = getrulewhatsit(line, wd, ht, dp)
704         local pack = box.id == node.id"hlist" and node.hpack or node.vpack
705         local list = pack(pl, glue, "exactly")
706         head = node.insert_after(head, curr, list)
707         head, curr = node.remove(head, curr)
708       end
709     elseif curr.id == node.id"rule" and curr.subtype == 0 then
710       local line = getemboldenwidth(curr, fakebold)
711       local wd,ht,dp = getrulemetric(box, curr)
712       if box.id == node.id"vlist" then
713         ht, dp = 0, ht+dp
714       end
715       local pl = getrulewhatsit(line, wd, ht, dp)
716       local list
717       if box.id == node.id"hlist" then
718         list = node.hpack(pl, wd, "exactly")
719       else
720         list = node.vpack(pl, ht+dp, "exactly")
721       end
722       head = node.insert_after(head, curr, list)
723       head, curr = node.remove(head, curr)

```

```

724 elseif curr.id == node.id"glyph" and curr.font > 0 then
725     local f = curr.font
726     local key = format("%s:%s",f,fakebold)
727     local i = emboldenfonts[key]
728     if not i then
729         local ft = font.getfont(f) or font.getcopy(f)
730         if pdfmode then
731             width = ft.size * fakebold / factor * 10
732             emboldenfonts.width = width
733             ft.mode, ft.width = 2, width
734             i = font.define(ft)
735         else
736             if ft.format ~= "opentype" and ft.format ~= "truetype" then
737                 goto skip_type1
738             end
739             local name = ft.name:gsub("'",''):gsub(';','$','')
740             name = format('%s;embolden=%s;',name,fakebold)
741             _, i = fonts.constructors.readanddefine(name,ft.size)
742         end
743         emboldenfonts[key] = i
744     end
745     curr.font = i
746 end
747 ::skip_type1::
748 curr = node.getnext(curr)
749 end
750 return head
751 end
752 local function graphictextcolor (col, filldraw)
753 if col:find"^[%d%.:]+ $" then
754     col = col:explode":"
755     for i=1,#col do
756         col[i] = format("%.3f", col[i])
757     end
758     if pdfmode then
759         local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
760         col[#col+1] = filldraw == "fill" and op or op:upper()
761         return tableconcat(col, " ")
762     end
763     return format("[%s]", tableconcat(col, " "))
764 end
765 col = process_color(col):match"mpliboverridecolor=(.+)"
766 if pdfmode then
767     local t, tt = col:explode(), { }
768     local b = filldraw == "fill" and 1 or #t/2+1
769     local e = b == 1 and #t/2 or #t
770     for i=b,e do
771         tt[#tt+1] = t[i]
772     end
773     return tableconcat(tt, " ")
774 end
775 return col:gsub("^.- ", "")
776 end
777 luampplib.graphicstext = function (text, fakebold, fc, dc)

```



```

778 local fmt = process_tex_text(text):sub(1,-2)
779 local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
780 emboldenfonts.width = nil
781 local box = texgetbox(id)
782 box.head = embolden(box, box.head, fakebold)
783 local fill = graphictextcolor(fc,"fill")
784 local draw = graphictextcolor(dc,"draw")
785 local bc = pdfmode and "" or "pdf:bc "
786 return format('%s withprescript "mpliboverridecolor=%s%s %s"', fmt, bc, fill, draw)
787 end
788

```

#### luamplib's mplibglyph operator

```

789 local function mperr (str)
790   return format("hide(errmessage %q)", str)
791 end
792 local function getangle (a,b,c)
793   local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
794   if r > 180 then
795     r = r - 360
796   elseif r < -180 then
797     r = r + 360
798   end
799   return r
800 end
801 local function turning (t)
802   local r, n = 0, #t
803   for i=1,2 do
804     tableinsert(t, t[i])
805   end
806   for i=1,n do
807     r = r + getangle(t[i], t[i+1], t[i+2])
808   end
809   return r/360
810 end
811 local function glyphimage(t, fmt)
812   local q,p,r = {},{}
813   for i,v in ipairs(t) do
814     local cmd = v[#v]
815     if cmd == "m" then
816       p = {format('(%s,%s)',v[1],v[2])}
817       r = {{x=v[1],y=v[2]}}
818     else
819       local nt = t[i+1]
820       local last = not nt or nt[#nt] == "m"
821       if cmd == "l" then
822         local pt = t[i-1]
823         local seco = pt[#pt] == "m"
824         if (last or seco) and r[1].x == v[1] and r[1].y == v[2] then
825           else
826             tableinsert(p, format('--(%s,%s)',v[1],v[2]))
827             tableinsert(r, {x=v[1],y=v[2]})
828           end
829         if last then
830           tableinsert(p, '--cycle')
831         end
832       end
833     end
834   end
835 end

```

```

831     end
832 elseif cmd == "c" then
833     tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
834     if last and r[1].x == v[5] and r[1].y == v[6] then
835         tableinsert(p, '..cycle')
836     else
837         tableinsert(p, format('..(%s,%s)',v[5],v[6]))
838         if last then
839             tableinsert(p, '--cycle')
840         end
841         tableinsert(r, {x=v[5],y=v[6]})
842     end
843 else
844     return mperr"unknown operator"
845 end
846 if last then
847     tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
848 end
849 end
850 end
851 r = { }
852 if fmt == "opentype" then
853     for _,v in ipairs(q[1]) do
854         tableinsert(r, format('addto currentpicture contour %s;',v))
855     end
856     for _,v in ipairs(q[2]) do
857         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
858     end
859 else
860     for _,v in ipairs(q[2]) do
861         tableinsert(r, format('addto currentpicture contour %s;',v))
862     end
863     for _,v in ipairs(q[1]) do
864         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
865     end
866 end
867 return format('image(%s)', tableconcat(r))
868 end
869 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
870 function luamplib.glyph (f, c)
871     local filename, subfont, instance, kind, shapedata
872     local fid = tonumber(f) or font.id(f)
873     if fid > 0 then
874         local fontdata = font.getfont(fid) or font.getcopy(fid)
875         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
876         instance = fontdata.specification and fontdata.specification.instance
877         filename = filename and filename:gsub("^harfloaded:", "")
878     else
879         local name
880         f = f:match"^%s*(.)%s*$"
881         name, subfont, instance = f:match"^(.+)%%((d+)%)%%[(.-)]$"
882         if not name then
883             name, instance = f:match"^(.+)%%[(.-)]$" -- SourceHanSansK-VF.otf[Heavy]
884         end

```

```

885     if not name then
886         name, subfont = f:match"(.)%((%d+)%)$" -- Times.ttc(2)
887     end
888     name = name or f
889     subfont = (subfont or 0)+1
890     instance = instance and instance:lower()
891     for _,ftype in ipairs{"opentype", "truetype"} do
892         filename = kpse.find_file(name, ftype.." fonts")
893         if filename then
894             kind = ftype; break
895         end
896     end
897 end
898 if kind ~= "opentype" and kind ~= "truetype" then
899     f = fid and fid > 0 and tex.fontname(fid) or f
900     if kpse.find_file(f, "tfm") then
901         return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
902     else
903         return mperr"font not found"
904     end
905 end
906 local time = lfsattributes(filename,"modification")
907 local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
908 local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
909 local newname = format("%s/%s.lua", cachedir or outputdir, h)
910 local newtime = lfsattributes(newname,"modification") or 0
911 if time == newtime then
912     shapedata = require(newname)
913 end
914 if not shapedata then
915     shapedata = fonts and fonts.handlers.otf.readers.loadshapes(filename,subfont,instance)
916     if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
917     table.toFile(newname, shapedata, "return")
918     lfstouch(newname, time, time)
919 end
920 local gid = tonumber(c)
921 if not gid then
922     local uni = utf8.codepoint(c)
923     for i,v in pairs(shapedata.glyphs) do
924         if c == v.name or uni == v.unicode then
925             gid = i; break
926         end
927     end
928 end
929 if not gid then return mperr"cannot get GID (glyph id)" end
930 local fac = 1000 / (shapedata.units or 1000)
931 local t = shapedata.glyphs[gid].segments
932 if not t then return "image()" end
933 for i,v in ipairs(t) do
934     if type(v) == "table" then
935         for ii,vv in ipairs(v) do
936             if type(vv) == "number" then
937                 t[i][ii] = format("%.0f", vv * fac)
938             end
939         end
940     end
941 end

```

```

939     end
940   end
941 end
942 kind = shapedata.format or kind
943 return glyphimage(t, kind)
944 end
945
  mpliboutlinetext : based on mkiv's font-mps.lua
946 local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
947   unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
948 local outline_horz, outline_vert
949 function outline_vert (res, box, curr, xshift, yshift)
950   local b2u = box.dir == "LTL"
951   local dy = (b2u and -box.depth or box.height)/factor
952   local ody = dy
953   while curr do
954     if curr.id == node.id"rule" then
955       local wd, ht, dp = getrulemetric(box, curr, true)
956       local hd = ht + dp
957       if hd ~= 0 then
958         dy = dy + (b2u and dp or -ht)
959         if wd ~= 0 and curr.subtype == 0 then
960           res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
961         end
962         dy = dy + (b2u and ht or -dp)
963       end
964     elseif curr.id == node.id"glue" then
965       local vwidth = node.effective_glue(curr,box)/factor
966       if curr.leader then
967         local curr, kind = curr.leader, curr.subtype
968         if curr.id == node.id"rule" then
969           local wd = getrulemetric(box, curr, true)
970           if wd ~= 0 then
971             local hd = vwidth
972             local dy = dy + (b2u and 0 or -hd)
973             if hd ~= 0 and curr.subtype == 0 then
974               res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
975             end
976           end
977         elseif curr.head then
978           local hd = (curr.height + curr.depth)/factor
979           if hd <= vwidth then
980             local dy, n, iy = dy, 0, 0
981             if kind == 100 or kind == 103 then -- todo: gleaders
982               local ady = abs(ody - dy)
983               local ndy = math.ceil(ady / hd) * hd
984               local diff = ndy - ady
985               n = (vwidth-diff) // hd
986               dy = dy + (b2u and diff or -diff)
987             else
988               n = vwidth // hd
989               if kind == 101 then
990                 local side = vwidth % hd / 2
991                 dy = dy + (b2u and side or -side)

```

```

992         elseif kind == 102 then
993             iy = vwidth % hd / (n+1)
994             dy = dy + (b2u and iy or -iy)
995         end
996     end
997     dy = dy + (b2u and curr.depth or -curr.height)/factor
998     hd = b2u and hd or -hd
999     iy = b2u and iy or -iy
1000     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1001     for i=1,n do
1002         res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1003         dy = dy + hd + iy
1004     end
1005 end
1006 end
1007 end
1008 dy = dy + (b2u and vwidth or -vwidth)
1009 elseif curr.id == node.id"kern" then
1010     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1011 elseif curr.id == node.id"vlist" then
1012     dy = dy + (b2u and curr.depth or -curr.height)/factor
1013     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1014     dy = dy + (b2u and curr.height or -curr.depth)/factor
1015 elseif curr.id == node.id"hlist" then
1016     dy = dy + (b2u and curr.depth or -curr.height)/factor
1017     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1018     dy = dy + (b2u and curr.height or -curr.depth)/factor
1019 end
1020 curr = node.getnext(curr)
1021 end
1022 return res
1023 end
1024 function outline_horz (res, box, curr, xshift, yshift, discwd)
1025     local r2l = box.dir == "TRT"
1026     local dx = r2l and (discwd or box.width/factor) or 0
1027     local dirs = { { dir = r2l, dx = dx } }
1028     while curr do
1029         if curr.id == node.id"dir" then
1030             local sign, dir = curr.dir:match"(.)(...)"
1031             local level, newdir = curr.level, r2l
1032             if sign == "+" then
1033                 newdir = dir == "TRT"
1034                 if r2l ~= newdir then
1035                     local n = node.getnext(curr)
1036                     while n do
1037                         if n.id == node.id"dir" and n.level+1 == level then break end
1038                         n = node.getnext(n)
1039                     end
1040                     n = n or node.tail(curr)
1041                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1042                 end
1043                 dirs[level] = { dir = r2l, dx = dx }
1044             else
1045                 local level = level + 1

```

```

1046     newdir = dirs[level].dir
1047     if r2l ~= newdir then
1048         dx = dirs[level].dx
1049     end
1050 end
1051 r2l = newdir
1052 elseif curr.char and curr.font and curr.font > 0 then
1053     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1054     local gid = ft.characters[curr.char].index or curr.char
1055     local scale = ft.size / factor / 1000
1056     local slant = (ft.slant or 0)/1000
1057     local extend = (ft.extend or 1000)/1000
1058     local squeeze = (ft.squeeze or 1000)/1000
1059     local expand = 1 + (curr.expansion_factor or 0)/1000000
1060     local xscale = scale * extend * expand
1061     local yscale = scale * squeeze
1062     dx = dx - (r2l and curr.width/factor*expand or 0)
1063     local xpos = dx + xshift + (curr.xoffset or 0)/factor
1064     local ypos = yshift + (curr.yoffset or 0)/factor
1065     local vertical = ft.shared and ft.shared.features.vertical and "rotated 90" or ""
1066     if vertical ~= "" then -- luatexko
1067         for _,v in ipairs(ft.characters[curr.char].commands or { }) do
1068             if v[1] == "down" then
1069                 ypos = ypos - v[2] / factor
1070             elseif v[1] == "right" then
1071                 xpos = xpos + v[2] / factor
1072             else
1073                 break
1074             end
1075         end
1076     end
1077     local image
1078     if ft.format == "opentype" or ft.format == "truetype" then
1079         image = luampplib.glyph(curr.font, gid)
1080     else
1081         local name, scale = ft.name, 1
1082         local vf = font.read_vf(name, ft.size)
1083         if vf and vf.characters[gid] then
1084             local cmds = vf.characters[gid].commands or {}
1085             for _,v in ipairs(cmds) do
1086                 if v[1] == "char" then
1087                     gid = v[2]
1088                 elseif v[1] == "font" and vf.fonts[v[2]] then
1089                     name = vf.fonts[v[2]].name
1090                     scale = vf.fonts[v[2]].size / ft.size
1091                 end
1092             end
1093         end
1094         image = format("glyph %s of %q scaled %f", gid, name, scale)
1095     end
1096     res[#res+1] = format("mpliboutlinepic[%i]:=s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1097         #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1098     dx = dx + (r2l and 0 or curr.width/factor*expand)
1099 elseif curr.replace then

```

```

1100     local width = node.dimensions(curr.replace)/factor
1101     dx = dx - (r2l and width or 0)
1102     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1103     dx = dx + (r2l and 0 or width)
1104 elseif curr.id == node.id"rule" then
1105     local wd, ht, dp = getrulemetric(box, curr, true)
1106     if wd ~= 0 then
1107         local hd = ht + dp
1108         dx = dx - (r2l and wd or 0)
1109         if hd ~= 0 and curr.subtype == 0 then
1110             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1111         end
1112         dx = dx + (r2l and 0 or wd)
1113     end
1114 elseif curr.id == node.id"glue" then
1115     local width = node.effective_glue(curr, box)/factor
1116     dx = dx - (r2l and width or 0)
1117     if curr.leader then
1118         local curr, kind = curr.leader, curr.subtype
1119         if curr.id == node.id"rule" then
1120             local wd, ht, dp = getrulemetric(box, curr, true)
1121             local hd = ht + dp
1122             if hd ~= 0 then
1123                 wd = width
1124                 if wd ~= 0 and curr.subtype == 0 then
1125                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1126                 end
1127             end
1128         elseif curr.head then
1129             local wd = curr.width/factor
1130             if wd <= width then
1131                 local dx = r2l and dx+width or dx
1132                 local n, ix = 0, 0
1133                 if kind == 100 or kind == 103 then -- todo: gleaders
1134                     local adx = abs(dx-dirs[1].dx)
1135                     local ndx = math.ceil(adx / wd) * wd
1136                     local diff = ndx - adx
1137                     n = (width-diff) // wd
1138                     dx = dx + (r2l and -diff-wd or diff)
1139                 else
1140                     n = width // wd
1141                     if kind == 101 then
1142                         local side = width % wd / 2
1143                         dx = dx + (r2l and -side-wd or side)
1144                     elseif kind == 102 then
1145                         ix = width % wd / (n+1)
1146                         dx = dx + (r2l and -ix-wd or ix)
1147                     end
1148                 end
1149                 wd = r2l and -wd or wd
1150                 ix = r2l and -ix or ix
1151                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1152                 for i=1,n do
1153                     res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)

```

```

1154         dx = dx + wd + ix
1155     end
1156 end
1157 end
1158 end
1159 dx = dx + (r2l and 0 or width)
1160 elseif curr.id == node.id"kern" then
1161     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1162 elseif curr.id == node.id"math" then
1163     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1164 elseif curr.id == node.id"vlist" then
1165     dx = dx - (r2l and curr.width/factor or 0)
1166     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1167     dx = dx + (r2l and 0 or curr.width/factor)
1168 elseif curr.id == node.id"hlist" then
1169     dx = dx - (r2l and curr.width/factor or 0)
1170     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1171     dx = dx + (r2l and 0 or curr.width/factor)
1172 end
1173 curr = node.getnext(curr)
1174 end
1175 return res
1176 end
1177 function luamplib.outlinetext (text)
1178     local fmt = process_tex_text(text)
1179     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1180     local box = texgetbox(id)
1181     local res = outline_horz({ }, box, box.head, 0, 0)
1182     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1183     return tableconcat(res) .. format("mpliboutlinenum=%i;", #res)
1184 end
1185

```

#### Our METAPOST preambles

```

1186 luamplib.preambles = {
1187     mplibcode = [[
1188 texscriptmode := 2;
1189 def rawtexttext (expr t) = runscript("luamplibtext{"&t&"}") enddef;
1190 def mplibcolor (expr t) = runscript("luamplibcolor{"&t&"}") enddef;
1191 def mplibdimen (expr t) = runscript("luamplibdimen{"&t&"}") enddef;
1192 def VerbatimTeX (expr t) = runscript("luamplibverbtex{"&t&"}") enddef;
1193 if known context_mlib:
1194     defaultfont := "cmtt10";
1195     let infont = normalinfont;
1196     let fontsize = normalfontsize;
1197     vardef thelabel@#(expr p,z) =
1198         if string p :
1199             thelabel@#(p infont defaultfont scaled defaultscale,z)
1200         else :
1201             p shifted (z + labeloffset*mfun_laboff@# -
1202                 (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1203                 (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1204         fi
1205     enddef;
1206 else:

```



```

1207 vardef texttext@# (text t) = rawtexttext (t) enddef;
1208 def message expr t =
1209   if string t: runscript("mp.report[=["&t&"]=]") else: errmessage "Not a string" fi
1210 enddef;
1211 def withtransparency (expr a, t) =
1212   withprescript "tr_alternative=" & if numeric a: decimal fi a
1213   withprescript "tr_transparency=" & decimal t
1214 enddef;
1215 vardef ddecimal primary p =
1216   decimal xpart p & " " & decimal ypart p
1217 enddef;
1218 vardef boundingbox primary p =
1219   if (path p) or (picture p) :
1220     llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1221   else :
1222     origin
1223   fi -- cycle
1224 enddef;
1225 fi
1226 def resolvedcolor(expr s) =
1227   runscript("return luamplib.shadecolor('"&s &"')")
1228 enddef;
1229 def colordecimals primary c =
1230   if cmykcolor c:
1231     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1232     decimal yellowpart c & ":" & decimal blackpart c
1233   elseif rgbcolor c:
1234     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1235   elseif string c:
1236     if known graphictextpic: c else: colordecimals resolvedcolor(c) fi
1237   else:
1238     decimal c
1239   fi
1240 enddef;
1241 def externalfigure primary filename =
1242   draw rawtexttext("\includegraphics{"& filename &"}")
1243 enddef;
1244 def TEX = texttext enddef;
1245 def mplibtexcolor primary c =
1246   runscript("return luamplib.gettexcolor('"&c &"')")
1247 enddef;
1248 def mplibrbgtexcolor primary c =
1249   runscript("return luamplib.gettexcolor('"&c &"', 'rgb')")
1250 enddef;
1251 def mplibgraphictext primary t =
1252   begingroup;
1253   mplibgraphictext_ (t)
1254 enddef;
1255 def mplibgraphictext_ (expr t) text rest =
1256   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor,
1257   fb, fc, dc, graphictextpic;
1258   picture graphictextpic; graphictextpic := nullpicture;
1259   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1260   let scale = scaled;

```

```

1261 def fakebold primary c = hide(fb:=c;) enddef;
1262 def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1263 def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1264 let withfillcolor = fillcolor; let withdrawcolor = drawcolor;
1265 addto graphictextpic doublepath origin rest; graphictextpic:=nullpicture;
1266 def fakebold primary c = enddef;
1267 let fillcolor = fakebold; let drawcolor = fakebold;
1268 let withfillcolor = fillcolor; let withdrawcolor = drawcolor;
1269 image(draw runscript("return luamplib.graphictext([==["&t&"]==], "
1270   & decimal fb &"', '& fc &"', '& dc &'")) rest;))
1271 endgroup;
1272 enddef;
1273 def mplibglyph expr c of f =
1274   runscript (
1275     "return luamplib.glyph('"
1276       & if numeric f: decimal fi f
1277       & "'',"
1278       & if numeric c: decimal fi c
1279       & "'')")
1280   )
1281 enddef;
1282 def mplibdrawglyph expr g =
1283   draw image(
1284     save i; numeric i; i:=0;
1285     for item within g:
1286       i := i+1;
1287       fill pathpart item
1288       if i < length g: withpostscript "collect" fi;
1289     endfor
1290   )
1291 enddef;
1292 def mplib_do_outline_text_set_b (text f) (text d) text r =
1293   def mplib_do_outline_options_f = f enddef;
1294   def mplib_do_outline_options_d = d enddef;
1295   def mplib_do_outline_options_r = r enddef;
1296 enddef;
1297 def mplib_do_outline_text_set_f (text f) text r =
1298   def mplib_do_outline_options_f = f enddef;
1299   def mplib_do_outline_options_r = r enddef;
1300 enddef;
1301 def mplib_do_outline_text_set_u (text f) text r =
1302   def mplib_do_outline_options_f = f enddef;
1303 enddef;
1304 def mplib_do_outline_text_set_d (text d) text r =
1305   def mplib_do_outline_options_d = d enddef;
1306   def mplib_do_outline_options_r = r enddef;
1307 enddef;
1308 def mplib_do_outline_text_set_r (text d) (text f) text r =
1309   def mplib_do_outline_options_d = d enddef;
1310   def mplib_do_outline_options_f = f enddef;
1311   def mplib_do_outline_options_r = r enddef;
1312 enddef;
1313 def mplib_do_outline_text_set_n text r =
1314   def mplib_do_outline_options_r = r enddef;

```

```

1315 enddef;
1316 def mplib_do_outline_text_set_p = enddef;
1317 def mplib_fill_outline_text =
1318   for n=1 upto mpliboutlinenum:
1319     i:=0;
1320     for item within mpliboutlinepic[n]:
1321       i:=i+1;
1322       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1323       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostsript "collect"; fi
1324     endfor
1325   endfor
1326 enddef;
1327 def mplib_draw_outline_text =
1328   for n=1 upto mpliboutlinenum:
1329     for item within mpliboutlinepic[n]:
1330       draw pathpart item mplib_do_outline_options_d;
1331     endfor
1332   endfor
1333 enddef;
1334 def mplib_filldraw_outline_text =
1335   for n=1 upto mpliboutlinenum:
1336     i:=0;
1337     for item within mpliboutlinepic[n]:
1338       i:=i+1;
1339       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1340         fill pathpart item mplib_do_outline_options_f withpostsript "collect";
1341       else:
1342         draw pathpart item mplib_do_outline_options_f withpostsript "both";
1343       fi
1344     endfor
1345   endfor
1346 enddef;
1347 vardef mpliboutlinetext@# (expr t) text rest =
1348   save kind; string kind; kind := str @#;
1349   save i; numeric i;
1350   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1351   def mplib_do_outline_options_d = enddef;
1352   def mplib_do_outline_options_f = enddef;
1353   def mplib_do_outline_options_r = enddef;
1354   runscript("return luamplib.outlinetext[==["&t&"]==]");
1355   image ( addto currentpicture also image (
1356     if kind = "f":
1357       mplib_do_outline_text_set_f rest;
1358       mplib_fill_outline_text;
1359     elseif kind = "d":
1360       mplib_do_outline_text_set_d rest;
1361       mplib_draw_outline_text;
1362     elseif kind = "b":
1363       mplib_do_outline_text_set_b rest;
1364       mplib_fill_outline_text;
1365       mplib_draw_outline_text;
1366     elseif kind = "u":
1367       mplib_do_outline_text_set_u rest;
1368       mplib_filldraw_outline_text;

```

```

1369 elseif kind = "r":
1370     mplib_do_outline_text_set_r rest;
1371     mplib_draw_outline_text;
1372     mplib_fill_outline_text;
1373 elseif kind = "p":
1374     mplib_do_outline_text_set_p;
1375     mplib_draw_outline_text;
1376 else:
1377     mplib_do_outline_text_set_n rest;
1378     mplib_fill_outline_text;
1379 fi;
1380 ) mplib_do_outline_options_r; )
1381 enddef ;
1382 primarydef t withpattern p =
1383 image(
1384 if cycle t:
1385 fill
1386 else:
1387 draw
1388 fi
1389 t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1390 enddef;
1391 vardef mplibtransformmatrix (text e) =
1392 save t; transform t;
1393 t = identity e;
1394 runscript("luamplib.transformmatrix = {"
1395 & decimal xpart t & ","
1396 & decimal ypart t & ","
1397 & decimal xpart t & ","
1398 & decimal ypart t & ","
1399 & decimal xpart t & ","
1400 & decimal ypart t & ","
1401 & "}");
1402 enddef;
1403 primarydef p withfademethod s =
1404 if picture p:
1405 image(
1406 draw p;
1407 draw center p withprescript "mplibfadestate=stop";
1408 )
1409 else:
1410 p withprescript "mplibfadestate=stop"
1411 fi
1412 withprescript "mplibfadetype=" & s
1413 withprescript "mplibfadebbox=" &
1414 decimal (xpart llcorner p -1/4) & ":" &
1415 decimal (ypart llcorner p -1/4) & ":" &
1416 decimal (xpart urcorner p +1/4) & ":" &
1417 decimal (ypart urcorner p +1/4)
1418 enddef;
1419 def withfadeopacity (expr a,b) =
1420 withprescript "mplibfadeopacity=" &
1421 decimal a & ":" &
1422 decimal b

```

```

1423 enddef;
1424 def withfadevector (expr a,b) =
1425   withprescript "mplibfadevector=" &
1426     decimal xpart a & ":" &
1427     decimal ypart a & ":" &
1428     decimal xpart b & ":" &
1429     decimal ypart b
1430 enddef;
1431 let withfadecenter = withfadevector;
1432 def withfaderadius (expr a,b) =
1433   withprescript "mplibfaderadius=" &
1434     decimal a & ":" &
1435     decimal b
1436 enddef;
1437 def withfadebbox (expr a,b) =
1438   withprescript "mplibfadebbox=" &
1439     decimal xpart a & ":" &
1440     decimal ypart a & ":" &
1441     decimal xpart b & ":" &
1442     decimal ypart b
1443 enddef;
1444 primarydef p asgroup s =
1445   image(
1446     draw center p
1447       withprescript "mplibgroupbbox=" &
1448         decimal (xpart llcorner p -1/4) & ":" &
1449         decimal (ypart llcorner p -1/4) & ":" &
1450         decimal (xpart urcorner p +1/4) & ":" &
1451         decimal (ypart urcorner p +1/4)
1452       withprescript "gr_state=start"
1453       withprescript "gr_type=" & s;
1454     draw p;
1455     draw center p withprescript "gr_state=stop";
1456   )
1457 enddef;
1458 def withgroupbbox (expr a,b) =
1459   withprescript "mplibgroupbbox=" &
1460     decimal xpart a & ":" &
1461     decimal ypart a & ":" &
1462     decimal xpart b & ":" &
1463     decimal ypart b
1464 enddef;
1465 def withgroupname expr s =
1466   withprescript "mplibgroupname=" & s
1467 enddef;
1468 def usemplibgroup primary s =
1469   draw maketext("\csname luamplib.group." & s & "\endcsname")
1470   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1471 enddef;
1472 path    mplib_shade_path ;
1473 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1474 numeric mplib_shade_fx, mplib_shade_fy ;
1475 numeric mplib_shade_lx, mplib_shade_ly ;
1476 numeric mplib_shade_nx, mplib_shade_ny ;

```

```

1477 numeric mplib_shade_dx, mplib_shade_dy ;
1478 numeric mplib_shade_tx, mplib_shade_ty ;
1479 primarydef p withshadingmethod m =
1480   p
1481   if picture p :
1482     withprescript "sh_operand_type=picture"
1483     if textual p:
1484       withprescript "sh_transform=no"
1485       mplib_with_shade_method (boundingbox p, m)
1486     else:
1487       withprescript "sh_transform=yes"
1488       mplib_with_shade_method (pathpart p, m)
1489     fi
1490   else :
1491     withprescript "sh_transform=yes"
1492     mplib_with_shade_method (p, m)
1493   fi
1494 enddef;
1495 def mplib_with_shade_method (expr p, m) =
1496   hide(mplib_with_shade_method_analyze(p))
1497   withprescript "sh_domain=0 1"
1498   withprescript "sh_color=into"
1499   withprescript "sh_color_a=" & colordecimals white
1500   withprescript "sh_color_b=" & colordecimals black
1501   withprescript "sh_first=" & ddecimal point 0 of p
1502   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1503   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1504   if m = "linear" :
1505     withprescript "sh_type=linear"
1506     withprescript "sh_factor=1"
1507     withprescript "sh_center_a=" & ddecimal llcorner p
1508     withprescript "sh_center_b=" & ddecimal urcorner p
1509   else :
1510     withprescript "sh_type=circular"
1511     withprescript "sh_factor=1.2"
1512     withprescript "sh_center_a=" & ddecimal center p
1513     withprescript "sh_center_b=" & ddecimal center p
1514     withprescript "sh_radius_a=" & decimal 0
1515     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1516   fi
1517 enddef;
1518 def mplib_with_shade_method_analyze(expr p) =
1519   mplib_shade_path := p ;
1520   mplib_shade_step := 1 ;
1521   mplib_shade_fx := xpart point 0 of p ;
1522   mplib_shade_fy := ypart point 0 of p ;
1523   mplib_shade_lx := mplib_shade_fx ;
1524   mplib_shade_ly := mplib_shade_fy ;
1525   mplib_shade_nx := 0 ;
1526   mplib_shade_ny := 0 ;
1527   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1528   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1529   for i=1 upto length(p) :
1530     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;

```

```

1531   mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1532   if mplib_shade_tx > mplib_shade_dx :
1533       mplib_shade_nx := i + 1 ;
1534       mplib_shade_lx := xpart point i of p ;
1535       mplib_shade_dx := mplib_shade_tx ;
1536   fi ;
1537   if mplib_shade_ty > mplib_shade_dy :
1538       mplib_shade_ny := i + 1 ;
1539       mplib_shade_ly := ypart point i of p ;
1540       mplib_shade_dy := mplib_shade_ty ;
1541   fi ;
1542   endfor ;
1543   enddef;
1544   vardef mplib_max_radius(expr p) =
1545       max (
1546           (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1547           (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1548           (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1549           (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1550       )
1551   enddef;
1552   def withshadingstep (text t) =
1553       hide(mplib_shade_step := mplib_shade_step + 1 ;)
1554       withprescript "sh_step=" & decimal mplib_shade_step
1555       t
1556   enddef;
1557   def withshadingradius expr a =
1558       withprescript "sh_radius_a=" & decimal (xpart a)
1559       withprescript "sh_radius_b=" & decimal (ypart a)
1560   enddef;
1561   def withshadingorigin expr a =
1562       withprescript "sh_center_a=" & ddecimal a
1563       withprescript "sh_center_b=" & ddecimal a
1564   enddef;
1565   def withshadingvector expr a =
1566       withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1567       withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1568   enddef;
1569   def withshadingdirection expr a =
1570       withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1571       withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1572   enddef;
1573   def withshadingtransform expr a =
1574       withprescript "sh_transform=" & a
1575   enddef;
1576   def withshadingcenter expr a =
1577       withprescript "sh_center_a=" & ddecimal (
1578           center mplib_shade_path shifted (
1579               xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1580               ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1581           )
1582       )
1583   enddef;
1584   def withshadingdomain expr d =

```

```

1585 withprescript "sh_domain=" & ddecimal d
1586 enddef;
1587 def withshadingfactor expr f =
1588   withprescript "sh_factor=" & decimal f
1589 enddef;
1590 def withshadingfraction expr a =
1591   if mplib_shade_step > 0 :
1592     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1593   fi
1594 enddef;
1595 def withshadingcolors (expr a, b) =
1596   if mplib_shade_step > 0 :
1597     withprescript "sh_color=into"
1598     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1599     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1600   else :
1601     withprescript "sh_color=into"
1602     withprescript "sh_color_a=" & colordecimals a
1603     withprescript "sh_color_b=" & colordecimals b
1604   fi
1605 enddef;
1606 ],
1607 legacyverbatimtex = [[
1608 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1609 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1610 let VerbatimTeX = specialVerbatimTeX;
1611 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1612 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1613 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1614 "runscript(" &ditto&
1615 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1616 "luamplib.in_the_fig=false" &ditto& ");";
1617 ],
1618 texttextlabel = [[
1619 let luampliboriginalinfont = infont;
1620 primarydef s infont f =
1621   if (s < char 32)
1622     or (s = char 35) % #
1623     or (s = char 36) % $
1624     or (s = char 37) % %
1625     or (s = char 38) % &
1626     or (s = char 92) % \
1627     or (s = char 94) % ^
1628     or (s = char 95) % _
1629     or (s = char 123) % {
1630     or (s = char 125) % }
1631     or (s = char 126) % ~
1632     or (s = char 127) :
1633     s luampliboriginalinfont f
1634   else :
1635     rawtexttext(s)
1636   fi
1637 enddef;
1638 def fontsize expr f =

```



```

1639 begingroup
1640 save size; numeric size;
1641 size := mplibdimen("1em");
1642 if size = 0: 10pt else: size fi
1643 endgroup
1644 enddef;
1645 ]],
1646 }
1647

```

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

```

1648 luamplib.verbatiminput = false

```

Do not expand `btex ... etex`, `verbatimtex ... etex`, and string expressions.

```

1649 local function protect_expansion (str)
1650   if str then
1651     str = str:gsub("\\", "!!!Control!!!")
1652           :gsub("%%", "!!!Comment!!!")
1653           :gsub("#", "!!!HashSign!!!")
1654           :gsub("{", "!!!LBrace!!!")
1655           :gsub("}", "!!!RBrace!!!")
1656     return format("\\unexpanded{%s}", str)
1657   end
1658 end
1659 local function unprotect_expansion (str)
1660   if str then
1661     return str:gsub("!!!Control!!!", "\\")
1662           :gsub("!!!Comment!!!", "%")
1663           :gsub("!!!HashSign!!!", "#")
1664           :gsub("!!!LBrace!!!", "{")
1665           :gsub("!!!RBrace!!!", "}")
1666   end
1667 end
1668 luamplib.everymplib = setmetatable({ [""] = "" }, { __index = function(t) return t[""] end })
1669 luamplib.everyendmplib = setmetatable({ [""] = "" }, { __index = function(t) return t[""] end })
1670 function luamplib.process_mplibcode (data, instancename)
1671   texboxes.localid = 4096

```

This is needed for legacy behavior

```

1672 if luamplib.legacyverbatim then
1673   luamplib.figid, tex_code_pre_mplib = 1, {}
1674 end
1675 local everymplib = luamplib.everymplib[instancename]
1676 local everyendmplib = luamplib.everyendmplib[instancename]
1677 data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
1678 :gsub("\r", "\n")

```

These five lines are needed for `mplibverbatim` mode.

```

1679 if luamplib.verbatiminput then
1680   data = data:gsub("\\mpcolor%{.+}%b{", "mplibcolor(\"%1\")")
1681           :gsub("\\mpdim%{.+}%b{", "mplibdimen(\"%1\")")
1682           :gsub("\\mpdim%{.+}%a+", "mplibdimen(\"%1\")")
1683           :gsub(btex_etex, "btex %1 etex ")
1684           :gsub(verbatimtex_etex, "verbatimtex %1 etex;")

```

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.

```

1685 else
1686   data = data:gsub(btex_etex, function(str)
1687     return format("btex %s etex ", protect_expansion(str)) -- space
1688   end)
1689   :gsub(verbatimtex_etex, function(str)
1690     return format("verbatimtex %s etex;", protect_expansion(str)) -- semicolon
1691   end)
1692   :gsub("\".-\\\"", protect_expansion)
1693   :gsub("\\\\%", "\\0PerCent\\0")
1694   :gsub("%%. -\\n", "\\n")
1695   :gsub("%zPerCent%z", "\\%")
1696   run_tex_code(format("\\mplibtmptoks\\expandafter{\\expanded{\\%s}}", data))
1697   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

1698   :gsub("##", "#")
1699   :gsub("\".-\\\"", unprotect_expansion)
1700   :gsub(btex_etex, function(str)
1701     return format("btex %s etex", unprotect_expansion(str))
1702   end)
1703   :gsub(verbatimtex_etex, function(str)
1704     return format("verbatimtex %s etex", unprotect_expansion(str))
1705   end)
1706 end
1707 process(data, instancename)
1708 end
1709

```

For parsing prescript materials.

```

1710 local function script2table(s)
1711   local t = {}
1712   for _,i in ipairs(s:explode("\\13+")) do
1713     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
1714     if k and v and k ~= "" and not t[k] then
1715       t[k] = v
1716     end
1717   end
1718   return t
1719 end
1720

```

`pdf literals` will be stored in `figcontents` table, and written to pdf in one go at the end of the flushing figure. Subtable post is for the legacy behavior.

```

1721 local figcontents = { post = { } }
1722 local function put2output(a,...)
1723   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
1724 end
1725 local function pdf_startfigure(n,llx,lly,urx,ury)
1726   put2output("\\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
1727 end
1728 local function pdf_stopfigure()
1729   put2output("\\mplibstoptoPDF")
1730 end

```

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

```

1731 local function pdf_literalcode (...)
1732   put2output{ -2, format(...) :gsub(decimals,rmzeros) }
1733 end
1734 local start_pdf_code = pdfmode
1735   and function() pdf_literalcode"q" end
1736   or function() put2output"\special{pdf:bcontent}" end
1737 local stop_pdf_code = pdfmode
1738   and function() pdf_literalcode"Q" end
1739   or function() put2output"\special{pdf:econtent}" end
1740

```

Now we process hboxes created from btex ... etex or texttext(...) or TEX(...), all being the same internally.

```

1741 local function put_tex_boxes (object,prescript)
1742   local box = prescript.mplibtexboxid:explode":"
1743   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
1744   if n and tw and th then
1745     local op = object.path
1746     local first, second, fourth = op[1], op[2], op[4]
1747     local tx, ty = first.x_coord, first.y_coord
1748     local sx, rx, ry, sy = 1, 0, 0, 1
1749     if tw ~= 0 then
1750       sx = (second.x_coord - tx)/tw
1751       rx = (second.y_coord - ty)/tw
1752       if sx == 0 then sx = 0.00001 end
1753     end
1754     if th ~= 0 then
1755       sy = (fourth.y_coord - ty)/th
1756       ry = (fourth.x_coord - tx)/th
1757       if sy == 0 then sy = 0.00001 end
1758     end
1759     start_pdf_code()
1760     pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
1761     put2output("\mplibuttextbox{i}",n)
1762     stop_pdf_code()
1763   end
1764 end
1765

```

### Colors

```

1766 local prev_override_color
1767 local function do_preobj_CR(object,prescript)
1768   if object.postscript == "collect" then return end
1769   local override = prescript and prescript.mpliboverridecolor
1770   if override then
1771     if pdfmode then
1772       pdf_literalcode(override)
1773       override = nil
1774     else
1775       put2output("\special{%s}",override)
1776       prev_override_color = override
1777     end
1778   else

```

```

1779 local cs = object.color
1780 if cs and #cs > 0 then
1781     pdf_literalcode(luamplib.colorconverter(cs))
1782     prev_override_color = nil
1783 elseif not pdfmode then
1784     override = prev_override_color
1785     if override then
1786         put2output("\\special{%s}", override)
1787     end
1788 end
1789 end
1790 return override
1791 end
1792
    For transparency and shading
1793 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
1794 local pdfobjs, pdfetcs = {}, {}
1795 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
1796 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
1797 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
1798 local function update_pdfobjs (os, stream)
1799     local key = os
1800     if stream then key = key..stream end
1801     local on = key and pdfobjs[key]
1802     if on then
1803         return on,false
1804     end
1805     if pdfmode then
1806         if stream then
1807             on = pdf.immediateobj("stream",stream,os)
1808         elseif os then
1809             on = pdf.immediateobj(os)
1810         else
1811             on = pdf.reserveobj()
1812         end
1813     else
1814         on = pdfetcs.cnt or 1
1815         if stream then
1816             texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
1817         elseif os then
1818             texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
1819         else
1820             texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
1821         end
1822         pdfetcs.cnt = on + 1
1823     end
1824     if key then
1825         pdfobjs[key] = on
1826     end
1827     return on,true
1828 end
1829 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
1830 if pdfmode then
1831     pdfetcs.getpageres = pdf.getpageresources or function() return pdf.pageresources end

```

```

1832 local getpagers = pdfetcs.getpagers
1833 local setpagers = pdf.setpagersresources or function(s) pdf.pagersresources = s end
1834 local initialize_resources = function (name)
1835     local tabname = format("%s_res",name)
1836     pdfetcs[tabname] = { }
1837     if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
1838         local obj = pdf.reserveobj()
1839         setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
1840         luatexbase.add_to_callback("finish_pdffile", function()
1841             pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
1842         end,
1843         format("luamplib.%s.finish_pdffile",name))
1844     end
1845 end
1846 pdfetcs.fallback_update_resources = function (name, res)
1847     local tabname = format("%s_res",name)
1848     if not pdfetcs[tabname] then
1849         initialize_resources(name)
1850     end
1851     if luatexbase.callbacktypes.finish_pdffile then
1852         local t = pdfetcs[tabname]
1853         t[#t+1] = res
1854     else
1855         local tpr, n = getpagers() or "", 0
1856         tpr, n = tpr:gsub(format("/%s<<",name), "%1".res)
1857         if n == 0 then
1858             tpr = format("%s/%s<<%s>>", tpr, name, res)
1859         end
1860         setpagers(tpr)
1861     end
1862 end
1863 else
1864     texsprint {
1865         "\\luamplibatfirstshipout{",
1866         "\\special{pdf:obj @MPLibTr<<>>}",
1867         "\\special{pdf:obj @MPLibSh<<>>}",
1868         "\\special{pdf:obj @MPLibCS<<>>}",
1869         "\\special{pdf:obj @MPLibPt<<>>}}",
1870     }
1871 pdfetcs.resadded = { }
1872 pdfetcs.fallback_update_resources = function (name,res,obj)
1873     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
1874     if not pdfetcs.resadded[name] then
1875         texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
1876         pdfetcs.resadded[name] = obj
1877     end
1878 end
1879 end
1880

```

### Transparency

```

1881 local transparency_modes = { [0] = "Normal",
1882     "Normal",      "Multiply",      "Screen",      "Overlay",
1883     "SoftLight",   "HardLight",     "ColorDodge",  "ColorBurn",
1884     "Darken",      "Lighten",      "Difference",   "Exclusion",

```

```

1885 "Hue",          "Saturation", "Color",          "Luminosity",
1886 "Compatible",
1887 normal    = "Normal",    multiply = "Multiply",    screen = "Screen",
1888 overlay   = "Overlay",    softlight = "SoftLight",    hardlight = "HardLight",
1889 colordodge = "ColorDodge", colorburn = "ColorBurn",    darken = "Darken",
1890 lighten   = "Lighten",    difference = "Difference",    exclusion = "Exclusion",
1891 hue       = "Hue",        saturation = "Saturation",    color = "Color",
1892 luminosity = "Luminosity", compatible = "Compatible",
1893 }
1894 local function add_extgs_resources (on, new)
1895   local key = format("MPlibTr%s", on)
1896   if new then
1897     local val = format(pdfetcs.resfmt, on)
1898     if pdfmanagement then
1899       texsprint {
1900         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{", val, "}"
1901       }
1902     else
1903       local tr = format("/%s %s", key, val)
1904       if is_defined(pdfetcs.pgfbextgs) then
1905         texsprint { "\\csname ", pdfetcs.pgfbextgs, "\\endcsname{", tr, "}" }
1906       elseif is_defined"TRP@list" then
1907         texsprint(catat11,{
1908           [[\if@files\write\@auxout{]],
1909           [[\string\g@addto@macro\string\TRP@list{]],
1910           tr,
1911           [[}]]\fi]],
1912         })
1913         if not get_macro"TRP@list":find(tr) then
1914           texsprint(catat11,[[\global\TRP@reruntrue]])
1915         end
1916       else
1917         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPlibTr")
1918       end
1919     end
1920   end
1921   return key
1922 end
1923 local function do_preobj_TR(object,prescript)
1924   if object.postscript == "collect" then return end
1925   local opa = prescript and prescript.tr_transparency
1926   if opa then
1927     local key, on, os, new
1928     local mode = prescript.tr_alternative or 1
1929     mode = transparency_modes[tonumber(mode) or mode:lower()]
1930     if not mode then
1931       mode = prescript.tr_alternative
1932       warn("unsupported blend mode: '%s'", mode)
1933     end
1934     opa = format("%.3f", opa) :gsub(decimals,rmzeros)
1935     for i,v in ipairs{ {mode,opa},{ "Normal",1} } do
1936       os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[2])
1937       on, new = update_pdfobjs(os)
1938       key = add_extgs_resources(on,new)

```

```

1939     if i == 1 then
1940         pdf_literalcode("/%s gs",key)
1941     else
1942         return format("/%s gs",key)
1943     end
1944 end
1945 end
1946 end
1947

    Shading with metafun format.
1948 local function sh_pdfpageresources(shtype,domain,colorspace,ca,cb,coordinates,steps,fractions)
1949 for _,v in ipairs{ca,cb} do
1950     for i,vv in ipairs(v) do
1951         for ii,vvv in ipairs(vv) do
1952             v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
1953         end
1954     end
1955 end
1956 local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
1957 if steps > 1 then
1958     local list,bounds,encode = { },{ },{ }
1959     for i=1,steps do
1960         if i < steps then
1961             bounds[i] = format("%.3f", fractions[i] or 1)
1962         end
1963         encode[2*i-1] = 0
1964         encode[2*i] = 1
1965         os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
1966         :gsub(decimals,rmzeros)
1967         list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
1968     end
1969     os = tableconcat {
1970         "<</FunctionType 3",
1971         format("/Bounds[%s]", tableconcat(bounds, ' ')),
1972         format("/Encode[%s]", tableconcat(encode, ' ')),
1973         format("/Functions[%s]", tableconcat(list, ' ')),
1974         format("/Domain[%s]>>", domain),
1975     } :gsub(decimals,rmzeros)
1976 else
1977     os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
1978     :gsub(decimals,rmzeros)
1979 end
1980 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
1981 os = tableconcat {
1982     format("<</ShadingType %i", shtype),
1983     format("/ColorSpace %s", colorspace),
1984     format("/Function %s", objref),
1985     format("/Coords[%s]", coordinates),
1986     "/Extend[true true]/AntiAlias true>>",
1987 } :gsub(decimals,rmzeros)
1988 local on, new = update_pdfobjs(os)
1989 if new then
1990     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
1991     if pdfmanagement then

```

```

1992     texsprintf {
1993         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
1994     }
1995     else
1996         local res = format("/%s %s", key, val)
1997         pdfetcs.fallback_update_resources("Shading",res,"@MPLibSh")
1998     end
1999 end
2000 return on
2001 end
2002 local function color_normalize(ca,cb)
2003     if #cb == 1 then
2004         if #ca == 4 then
2005             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2006         else -- #ca = 3
2007             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2008         end
2009     elseif #cb == 3 then -- #ca == 4
2010         cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
2011     end
2012 end
2013 pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2014     run_tex_code({
2015         [[\color_model_new:nnn]],
2016         format("{mplibcolorspace_%s}", names:gsub(",","_")),
2017         format("{DeviceN}{names={%s}}", names),
2018         [[\edef\mplib@tempa{\pdf_object_ref_last:}]],
2019     }, ccexplat)
2020     local colorspace = get_macro'mplib@tempa'
2021     t[names] = colorspace
2022     return colorspace
2023 end })
2024 local function do_preobj_SH(object,prescript)
2025     local shade_no
2026     local sh_type = prescript and prescript.sh_type
2027     if not sh_type then
2028         return
2029     else
2030         local domain = prescript.sh_domain or "0 1"
2031         local centera = (prescript.sh_center_a or "0 0"):explode()
2032         local centerb = (prescript.sh_center_b or "0 0"):explode()
2033         local transform = prescript.sh_transform == "yes"
2034         local sx,sy,sr,dx,dy = 1,1,1,0,0
2035         if transform then
2036             local first = (prescript.sh_first or "0 0"):explode()
2037             local setx = (prescript.sh_set_x or "0 0"):explode()
2038             local sety = (prescript.sh_set_y or "0 0"):explode()
2039             local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2040             if x ~= 0 and y ~= 0 then
2041                 local path = object.path
2042                 local path1x = path[1].x_coord
2043                 local path1y = path[1].y_coord
2044                 local path2x = path[x].x_coord
2045                 local path2y = path[y].y_coord

```



```

2046     local dxa = path2x - path1x
2047     local dya = path2y - path1y
2048     local dx = setx[2] - first[1]
2049     local dy = sety[2] - first[2]
2050     if dxa ~= 0 and dya ~= 0 and dx ~= 0 and dy ~= 0 then
2051         sx = dxa / dx ; if sx < 0 then sx = - sx end
2052         sy = dya / dy ; if sy < 0 then sy = - sy end
2053         sr = math.sqrt(sx^2 + sy^2)
2054         dx = path1x - sx*first[1]
2055         dy = path1y - sy*first[2]
2056     end
2057 end
2058 end
2059 local ca, cb, colorspace, steps, fractions
2060 ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode":" }
2061 cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode":" }
2062 steps = tonumber(prescript.sh_step) or 1
2063 if steps > 1 then
2064     fractions = { prescript.sh_fraction_1 or 0 }
2065     for i=2,steps do
2066         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2067         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode":"
2068         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode":"
2069     end
2070 end
2071 if prescript.mplib_spotcolor then
2072     ca, cb = { }, { }
2073     local names, pos, objref = { }, -1, ""
2074     local script = object.prescript:explode"\13+"
2075     for i=#script,1,-1 do
2076         if script[i]:find"mplib_spotcolor" then
2077             local t, name, value = script[i]:explode"="[2]:explode":"
2078             value, objref, name = t[1], t[2], t[3]
2079             if not names[name] then
2080                 pos = pos+1
2081                 names[name] = pos
2082                 names[#names+1] = name
2083             end
2084             t = { }
2085             for j=1,names[name] do t[#t+1] = 0 end
2086             t[#t+1] = value
2087             tableinsert(#ca == #cb and ca or cb, t)
2088         end
2089     end
2090     for _,t in ipairs{ca,cb} do
2091         for _,tt in ipairs(t) do
2092             for i=1,#names-#tt do tt[#tt+1] = 0 end
2093         end
2094     end
2095     if #names == 1 then
2096         colorspace = objref
2097     else
2098         colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2099     end

```

```

2100 else
2101     local model = 0
2102     for _,t in ipairs{ca,cb} do
2103         for _,tt in ipairs(t) do
2104             model = model > #tt and model or #tt
2105         end
2106     end
2107     for _,t in ipairs{ca,cb} do
2108         for _,tt in ipairs(t) do
2109             if #tt < model then
2110                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2111             end
2112         end
2113     end
2114     colorspace = model == 4 and "/DeviceCMYK"
2115                 or model == 3 and "/DeviceRGB"
2116                 or model == 1 and "/DeviceGray"
2117                 or err"unknown color model"
2118 end
2119 if sh_type == "linear" then
2120     local coordinates = format("%f %f %f %f",
2121         dx + sx*centera[1], dy + sy*centera[2],
2122         dx + sx*centerb[1], dy + sy*centerb[2])
2123     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions)
2124 elseif sh_type == "circular" then
2125     local factor = prescript.sh_factor or 1
2126     local radiusa = factor * prescript.sh_radius_a
2127     local radiusb = factor * prescript.sh_radius_b
2128     local coordinates = format("%f %f %f %f %f %f",
2129         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2130         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2131     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions)
2132 else
2133     err"unknown shading type"
2134 end
2135 end
2136 return shade_no
2137 end
2138

```

Shading Patterns: much similar to the metafun's shade, but we can apply shading to textual pictures as well as paths.

```

2139 local function add_pattern_resources (key, val)
2140     if pdfmanagement then
2141         texsprint {
2142             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2143         }
2144     else
2145         local res = format("/%s %s", key, val)
2146         if is_defined(pdfetcs.pgfpattern) then
2147             texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2148         else
2149             pdfetcs.fallback_update_resources("Pattern",res,"@MPlibPt")
2150         end
2151     end
2152 end

```

```

2151 end
2152 end
2153 function luamplib.dolatelua (on, os)
2154   local h, v = pdf.getpos()
2155   h = format("%f", h/factor) :gsub(decimals,rmzeros)
2156   v = format("%f", v/factor) :gsub(decimals,rmzeros)
2157   if pdfmode then
2158     pdf.obj(on, format("<<%s/Matrix[1 0 0 1 %s %s]>>", os, h, v))
2159     pdf.refobj(on)
2160   else
2161     local shift = os:explode()
2162     if tonumber(h) ~= tonumber(shift[1]) or tonumber(v) ~= tonumber(shift[2]) then
2163       warn([[Add 'withprescript "sh_matrixshift=%s %s"' to the picture shading]], h, v)
2164     end
2165   end
2166 end
2167 local function do_preobj_shading (object, prescript)
2168   if not prescript or not prescript.sh_operand_type then return end
2169   local on = do_preobj_SH(object, prescript)
2170   os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2171   on = update_pdfobjs()
2172   if pdfmode then
2173     put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(\"on,\"[[\",os,\"]]) }" })
2174   else

```

Why @xpos @ypos do not work properly???

Anyway, this seems to be needed for proper functioning:

```

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

2175   if is_defined"RecordProperties" then
2176     put2output(tableconcat{
2177       "\\csname tex_savepos:D\\endcsname\\RecordProperties{luamplib/getpos/",on,"}{xpos,ypos}\\z
2178       \\special{pdf:put @mplibpdfobj",on," <<\",os,\"/Matrix[1 0 0 1 \\z
2179       \\csname dim_to_decimal_in_bp:n\\endcsname{\\RefProperty{luamplib/getpos/",on,"}{xpos}sp} \\z
2180       \\csname dim_to_decimal_in_bp:n\\endcsname{\\RefProperty{luamplib/getpos/",on,"}{ypos}sp}\\z
2181       ]>>}"
2182     })
2183   else
2184     local shift = prescript.sh_matrixshift or "0 0"
2185     textsprint{ "\\special{pdf:put @mplibpdfobj",on," <<\",os,\"/Matrix[1 0 0 1 \",shift,\"]>>}" }
2186     put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(\"on,\"[[\",shift,\"]]) }" })
2187   end
2188 end
2189 local key, val = format("MPLibPt%s", on), format(pdfetcs.resfmt, on)
2190 add_pattern_resources(key,val)
2191 pdf_literalcode("/Pattern cs/%s scn", key)

```

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

```

2192 prescript.sh_type = nil
2193 end
2194

```

Tiling Patterns

```

2195 pdfetcs.patterns = { }
2196 local function gather_resources (optres)
2197   local t, do_pattern = { }, not optres
2198   local names = {"ExtGState", "ColorSpace", "Shading"}
2199   if do_pattern then
2200     names[#names+1] = "Pattern"
2201   end
2202   if pdfmode then
2203     if pdfmanagement then
2204       for _,v in ipairs(names) do
2205         local pp = get_macro(format("g__pdfdict_/g__pdf_Core/Page/Resources/%s_prop",v))
2206         if pp and pp:find "__prop_pair" then
2207           t[#t+1] = format("/%s %s 0 R", v, ltx.pdf.object_id("__pdf/Page/Resources/"..v))
2208         end
2209       end
2210     else
2211       local res = pdfetcs.getpages() or ""
2212       run_tex_code[["\mplibtmptoks\expandafter{\the\pdfvariable pageresources}]]
2213       res = res .. texgettoks'mplibtmptoks'
2214       if do_pattern then return res end
2215       res = res:explode"/+"
2216       for _,v in ipairs(res) do
2217         v = v:match"^%s*(.)%s*$"
2218         if not v:find"Pattern" and not optres:find(v) then
2219           t[#t+1] = "/" .. v
2220         end
2221       end
2222     end
2223   else
2224     if pdfmanagement then
2225       for _,v in ipairs(names) do
2226         local pp = get_macro(format("g__pdfdict_/g__pdf_Core/Page/Resources/%s_prop",v))
2227         if pp and pp:find "__prop_pair" then
2228           run_tex_code {
2229             "\mplibtmptoks\expanded{{" ,
2230             format("/%s \\\csname pdf_object_ref:n\\endcsname{__pdf/Page/Resources/%s}",v,v),
2231             "}}",
2232           }
2233           t[#t+1] = texgettoks'mplibtmptoks'
2234         end
2235       end
2236     elseif is_defined(pdfetcs.pgfgextgs) then
2237       run_tex_code {
2238         "\mplibtmptoks\expanded{{" ,
2239         "\\\ifpgf@sys@pdf@extgs@exists /ExtGState @pgfgextgs\\fi",
2240         "\\\ifpgf@sys@pdf@colorspaces@exists /ColorSpace @pgfcolorspaces\\fi",
2241         do_pattern and "\\\ifpgf@sys@pdf@patterns@exists /Pattern @pgfpatterns \\fi" or "",
2242         "}}",
2243       }, catat11)
2244       t[#t+1] = texgettoks'mplibtmptoks'
2245     else
2246       for _,v in ipairs(names) do
2247         local vv = pdfetcs.resadded[v]
2248         if vv then

```

```

2249         t[#t+1] = format("/%s %s", v, vv)
2250     end
2251 end
2252 end
2253 end
2254 return tableconcat(t)
2255 end
2256 function luamplib.registerpattern ( boxid, name, opts )
2257     local box = texgetbox(boxid)
2258     local wd = format("%.3f",box.width/factor)
2259     local hd = format("%.3f", (box.height+box.depth)/factor)
2260     info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2261     if opts.xstep == 0 then opts.xstep = nil end
2262     if opts.ystep == 0 then opts.ystep = nil end
2263     if opts.colored == nil then
2264         opts.colored = opts.coloured
2265         if opts.colored == nil then
2266             opts.colored = true
2267         end
2268     end
2269     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2270     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2271     if opts.matrix and opts.matrix:find"%a" then
2272         local data = format("mplibtransformmatrix(%s);",opts.matrix)
2273         process(data,"@mplibtransformmatrix")
2274         local t = luamplib.transformmatrix
2275         opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2276         opts.xshift = opts.xshift or format("%f",t[5])
2277         opts.yshift = opts.yshift or format("%f",t[6])
2278     end
2279     local attr = {
2280         "/Type/Pattern",
2281         "/PatternType 1",
2282         format("/PaintType %i", opts.colored and 1 or 2),
2283         "/TilingType 2",
2284         format("/XStep %s", opts.xstep or wd),
2285         format("/YStep %s", opts.ystep or hd),
2286         format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2287     }
2288     local optres = opts.resources or ""
2289     optres = optres .. gather_resources(optres)
2290     local patterns = pdfetcs.patterns
2291     if pdfmode then
2292         if opts.bbox then
2293             attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2294         end
2295         attr = tableconcat(attr) :gsub(decimals,rmzeros)
2296         local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2297         patterns[name] = { id = index, colored = opts.colored }
2298     else
2299         local cnt = #patterns + 1
2300         local objname = "@mplibpattern" .. cnt
2301         local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2302         texpstr {

```

```

2303     "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2304     "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2305     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2306     "\\special{pdf:bcontent}",
2307     "\\special{pdf:bxobj ", objname, " ", metric, "}",
2308     "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2309     "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2310     "\\special{pdf:put @resources <<", optres, ">>}",
2311     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2312     "\\special{pdf:econtent}}",
2313   }
2314   patterns[cnt] = objname
2315   patterns[name] = { id = cnt, colored = opts.colored }
2316 end
2317 end
2318 local function pattern_colorspace (cs)
2319   local on, new = update_pdfobjs(format("/Pattern %s]", cs))
2320   if new then
2321     local key, val = format("MPLibCS%i", on), format(pdfetcs.resfmt, on)
2322     if pdfmanagement then
2323       texsprint {
2324         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{" , val, "}"
2325       }
2326     else
2327       local res = format("/%s %s", key, val)
2328       if is_defined(pdfetcs.pgfcOLORSPACE) then
2329         texsprint { "\\csname ", pdfetcs.pgfcOLORSPACE, "\\endcsname{" , res, "}" }
2330       else
2331         pdfetcs.fallback_update_resources("ColorSpace", res, "@MPLibCS")
2332       end
2333     end
2334   end
2335   return on
2336 end
2337 local function do_preobj_PAT(object, prescript)
2338   local name = prescript and prescript.mplibpattern
2339   if not name then return end
2340   local patterns = pdfetcs.patterns
2341   local patt = patterns[name]
2342   local index = patt and patt.id or err("cannot get pattern object '%s'", name)
2343   local key = format("MPLibPt%s", index)
2344   if patt.colored then
2345     pdf_literalcode("/Pattern cs /%s scn", key)
2346   else
2347     local color = prescript.mpliboverridecolor
2348     if not color then
2349       local t = object.color
2350       color = t and #t>0 and luamplib.colorconverter(t)
2351     end
2352     if not color then return end
2353     local cs
2354     if color:find" cs " or color:find"@pdf.obj" then
2355       local t = color:explode()
2356       if pdfmode then

```

```

2357         cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
2358         color = t[3]
2359     else
2360         cs = t[2]
2361         color = t[3]:match"%[(.+)%"
2362     end
2363 else
2364     local t = colorsplit(color)
2365     cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
2366     color = tableconcat(t, " ")
2367 end
2368 pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
2369 end
2370 if not patt.done then
2371     local val = pdfmode and format("%s 0 R",index) or patterns[index]
2372     add_pattern_resources(key,val)
2373 end
2374 patt.done = true
2375 end
2376

```

### Fading

```

2377 pdfetcs.fading = { }
2378 local function do_preobj_FADE (object, prescript)
2379     local fd_type = prescript and prescript.mplibfadetype
2380     local fd_stop = prescript and prescript.mplibfadestate
2381     if not fd_type then
2382         return fd_stop -- returns "stop" (if picture) or nil
2383     end
2384     local bbox = prescript.mplibfadebbox:explode":"
2385     local dx, dy = -bbox[1], -bbox[2]
2386     local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
2387     if not vec then
2388         if fd_type == "linear" then
2389             vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
2390         else
2391             local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
2392             vec = {centerx, centery, centerx, centery} -- center for both circles
2393         end
2394     end
2395     local coords = { vec[1]+dx, vec[2]+dy, vec[3]+dx, vec[4]+dy }
2396     if fd_type == "linear" then
2397         coords = format("%f %f %f %f", tableunpack(coords))
2398     elseif fd_type == "circular" then
2399         local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
2400         local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":"
2401         tableinsert(coords, 3, radius[1])
2402         tableinsert(coords, radius[2])
2403         coords = format("%f %f %f %f %f %f", tableunpack(coords))
2404     else
2405         err("unknown fading method '%s'", fd_type)
2406     end
2407     fd_type = fd_type == "linear" and 2 or 3
2408     local opa = (prescript.mplibfadeopacity or "1:0"):explode":"
2409     local on, os, new

```

```

2410 on = sh_pdfpageresources(fd_type, "0 1", "/DeviceGray", {{opaq[1]}}, {{opaq[2]}}, coords, 1)
2411 os = format("<</PatternType 2/Shading %s>>", format(pdfetcs.resfmt, on))
2412 on = update_pdfobjs(os)
2413 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
2414 local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
2415 :gsub(decimals,rmzeros)
2416 os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
2417 on = update_pdfobjs(os)
2418 local resources = format(pdfetcs.resfmt, on)
2419 on = update_pdfobjs"<</S/Transparency/CS/DeviceGray>>"
2420 local attr = tableconcat{
2421     "/Subtype/Form",
2422     "/BBox[" .. bbox .. "]",
2423     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
2424     "/Resources " .. resources,
2425     "/Group " .. format(pdfetcs.resfmt, on),
2426 } :gsub(decimals,rmzeros)
2427 on = update_pdfobjs(attr, streamtext)
2428 os = "<</S/Transparency/CS/DeviceGray>>" .. format(pdfetcs.resfmt, on) .. ">>>>"
2429 on, new = update_pdfobjs(os)
2430 local key = add_extgs_resources(on,new)
2431 start_pdf_code()
2432 pdf_literalcode("/%s gs", key)
2433 if fd_stop then return "standalone" end
2434 return "start"
2435 end
2436

```

#### Transparency Group

```

2437 pdfetcs.tr_group = { shifts = { } }
2438 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
2439 local function do_preobj_GRP (object, prescript)
2440     local grstate = prescript and prescript.gr_state
2441     if not grstate then return end
2442     local trgroup = pdfetcs.tr_group
2443     if grstate == "start" then
2444         trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
2445         trgroup.isolated, trgroup.knockout = false, false
2446         for _,v in ipairs(prescript.gr_type:explode",") do
2447             trgroup[v] = true
2448         end
2449         trgroup.bbox = prescript.mplibgroupbbox:explode":"
2450         put2output[["\begingroup\setbox\mplibscratchbox\hbox\bgroup]]
2451     elseif grstate == "stop" then
2452         local llx,lly,urx,ury = tableunpack(trgroup.bbox)
2453         put2output(tableconcat{
2454             "\egroup",
2455             format("\wd\mplibscratchbox %fbp", urx-llx),
2456             format("\ht\mplibscratchbox %fbp", ury-lly),
2457             "\dp\mplibscratchbox 0pt",
2458         })
2459         local grattr = format("/Group<</S/Transparency/I %s/K %s>>",trgroup.isolated,trgroup.knockout)
2460         local res = gather_resources()
2461         local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
2462         if pdfmode then

```



```

2463 put2output(tableconcat{
2464     "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
2465     "/BBox[" .. bbox .. "]", grattr .. "}" resources{ .. res .. "}" \\mplibscratchbox",
2466     "\\luamplibtagasgroupbegin",
2467     "[[setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}]]",
2468     "[[\\wd\\mplibscratchbox 0pt\\ht\\mplibscratchbox 0pt\\dp\\mplibscratchbox 0pt]]",
2469     "[[box\\mplibscratchbox]]",
2470     "\\luamplibtagasgroupend",
2471     "\\endgroup",
2472     "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{",
2473     "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
2474     "\\useboxresource \\the\\lastsavedboxresourceindex",
2475     "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
2476     "\\box\\mplibscratchbox}",
2477 })
2478 else
2479     trgroup.cnt = (trgroup.cnt or 0) + 1
2480     local objname = format("@mplibtrgr%s", trgroup.cnt)
2481     put2output(tableconcat{
2482         "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
2483         "\\unhbox\\mplibscratchbox",
2484         "\\special{pdf:put @resources <<", res, ">>}",
2485         "\\special{pdf:exobj <<", grattr, ">>}",
2486         "\\special{pdf:uxobj ", objname, "}",
2487         "\\endgroup",
2488     })
2489     token.set_macro("luamplib.group." .. trgroup.name, tableconcat{
2490         "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
2491         "\\special{pdf:uxobj ", objname, "}",
2492         "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
2493         "\\box\\mplibscratchbox",
2494     }, "global")
2495     end
2496     trgroup.shifts[trgroup.name] = { llx, lly }
2497 end
2498 return grstate
2499 end
2500 function luamplib.registergroup (boxid, name, opts)
2501     local box = texgetbox(boxid)
2502     local wd, ht, dp = node.getwhd(box)
2503     local res = (opts.resources or "") .. gather_resources()
2504     local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
2505     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2506     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2507     if opts.matrix and opts.matrix:find"%a" then
2508         local data = format("mplibtransformmatrix(%s);",opts.matrix)
2509         process(data,"@mplibtransformmatrix")
2510         opts.matrix = format("%f %f %f %f %f %f",tableunpack(luamplib.transformmatrix))
2511     end
2512     local grtype = 3
2513     if opts.bbox then
2514         attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2515         grtype = 2
2516     end

```

```

2517 if opts.matrix then
2518   attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
2519   grtype = opts.bbox and 4 or 1
2520 end
2521 if opts.asgroup then
2522   local t = { isolated = false, knockout = false }
2523   for _,v in ipairs(opts.asgroup:explode",+") do t[v] = true end
2524   attr[#attr+1] = format("/Group<</S/Transparency/I %s/K %s>>", t.isolated, t.knockout)
2525 end
2526 local trgroup = pdfetcs.tr_group
2527 trgroup.shifts[name] = { get_macro'MPlx', get_macro'MPlly' }
2528 local whd
2529 if pdfmode then
2530   attr = tableconcat(attr) :gsub(decimals,rmzeros)
2531   local index = tex.saveboxresource(boxid, attr, res, true, grtype)
2532   token.set_macro("luamplib.group"..name, tableconcat{
2533     "\\useboxresource ", index,
2534     }, "global")
2535   whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
2536 else
2537   trgroup.cnt = (trgroup.cnt or 0) + 1
2538   local objname = format("@mplibtrgr%s", trgroup.cnt)
2539   textsprint {
2540     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2541     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2542     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2543     "\\special{pdf:bcontent}",
2544     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
2545     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2546     "\\special{pdf:put @resources <<, res, ">>}",
2547     "\\special{pdf:exobj <<, tableconcat(attr), ">>}",
2548     "\\special{pdf:econtent}}",
2549   }
2550   token.set_macro("luamplib.group"..name, tableconcat{
2551     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
2552     "\\wd\\mplibscratchbox ", wd, "sp",
2553     "\\ht\\mplibscratchbox ", ht, "sp",
2554     "\\dp\\mplibscratchbox ", dp, "sp",
2555     "\\box\\mplibscratchbox",
2556     }, "global")
2557   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
2558 end
2559 info("w/h/d of group '%s': %s", name, whd)
2560 end
2561
2562 local function stop_special_effects(fade,opaq,over)
2563   if fade then -- fading
2564     stop_pdf_code()
2565   end
2566   if opaq then -- opacity
2567     pdf_literalcode(opaq)
2568   end
2569   if over then -- color
2570     put2output "\\special{pdf:ec}"

```

```

2571 end
2572 end
2573
    Codes below for inserting PDF literals are mostly from ConTeXt general, with small
    changes when needed.

2574 local function getobjects(result,figure,f)
2575     return figure:objects()
2576 end
2577
2578 function luamplib.convert (result, flusher)
2579     luamplib.flush(result, flusher)
2580     return true -- done
2581 end
2582
2583 local function pdf_textfigure(font,size,text,width,height,depth)
2584     text = text:gsub(".",function(c)
2585         return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
2586     end)
2587     put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
2588 end
2589
2590 local bend_tolerance = 131/65536
2591
2592 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
2593
2594 local function pen_characteristics(object)
2595     local t = mplib.pen_info(object)
2596     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
2597     divider = sx*sy - rx*ry
2598     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
2599 end
2600
2601 local function concat(px, py) -- no tx, ty here
2602     return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
2603 end
2604
2605 local function curved(ith,pth)
2606     local d = pth.left_x - ith.right_x
2607     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
2608         d = pth.left_y - ith.right_y
2609         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
2610             return false
2611         end
2612     end
2613     return true
2614 end
2615
2616 local function flushnormalpath(path,open)
2617     local pth, ith
2618     for i=1,#path do
2619         pth = path[i]
2620         if not ith then
2621             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)

```

```

2622 elseif curved(ith,pth) then
2623   pdf_literalcode("%f %f %f %f %f %f c",ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
2624 else
2625   pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
2626 end
2627 ith = pth
2628 end
2629 if not open then
2630   local one = path[1]
2631   if curved(pth,one) then
2632     pdf_literalcode("%f %f %f %f %f %f c",pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
2633   else
2634     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2635   end
2636 elseif #path == 1 then -- special case .. draw point
2637   local one = path[1]
2638   pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2639 end
2640 end
2641
2642 local function flushconcatpath(path,open)
2643 pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
2644 local pth, ith
2645 for i=1,#path do
2646   pth = path[i]
2647   if not ith then
2648     pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
2649   elseif curved(ith,pth) then
2650     local a, b = concat(ith.right_x,ith.right_y)
2651     local c, d = concat(pth.left_x,pth.left_y)
2652     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
2653   else
2654     pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
2655   end
2656   ith = pth
2657 end
2658 if not open then
2659   local one = path[1]
2660   if curved(pth,one) then
2661     local a, b = concat(pth.right_x,pth.right_y)
2662     local c, d = concat(one.left_x,one.left_y)
2663     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
2664   else
2665     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2666   end
2667 elseif #path == 1 then -- special case .. draw point
2668   local one = path[1]
2669   pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2670 end
2671 end
2672

```

Finally, flush figures by inserting PDF literals.

```

2673 function luamplib.flush (result,flusher)
2674   if result then

```

```

2675 local figures = result.fig
2676 if figures then
2677   for f=1, #figures do
2678     info("flushing figure %s",f)
2679     local figure = figures[f]
2680     local objects = getobjects(result,figure,f)
2681     local fignum = tonumber(figure:filename():match("[%d]+")) or figure:charcode() or 0
2682     local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2683     local bbox = figure:boundingbox()
2684     local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
2685     if urx < llx then

```

luamplib silently ignores this invalid figure for those that do not contain `beginfig ... endfig`. (issue #70) Original code of ConTeXt general was:

```

-- invalid
pdf_startfigure(fignum,0,0,0,0)
pdf_stopfigure()

2686 else

```

For legacy behavior, insert ‘pre-fig’  $\TeX$  code here.

```

2687 if tex_code_pre_mplib[f] then
2688   put2output(tex_code_pre_mplib[f])
2689 end
2690 pdf_startfigure(fignum,llx,lly,urx,ury)
2691 start_pdf_code()
2692 if objects then
2693   local savedpath = nil
2694   local savedhtap = nil
2695   for o=1,#objects do
2696     local object      = objects[o]
2697     local objecttype  = object.type

```

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

```

2698 local prescript      = object.prescript
2699 prescript = prescript and script2table(prescript) -- prescript is now a table
2700 local cr_over = do_preobj_CR(object,prescript) -- color
2701 local tr_opaq = do_preobj_TR(object,prescript) -- opacity
2702 local fading_ = do_preobj_FADE(object,prescript) -- fading
2703 local trgroup = do_preobj_GRP(object,prescript) -- transparency group
2704 local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
2705 local shading_ = do_preobj_shading(object,prescript) -- shading pattern
2706 if prescript and prescript.mplibtexboxid then
2707   put_tex_boxes(object,prescript)
2708 elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
2709 elseif objecttype == "start_clip" then
2710   local evenodd = not object.istext and object.postscript == "evenodd"
2711   start_pdf_code()
2712   flushnormalpath(object.path,false)
2713   pdf_literalcode(evenodd and "W* n" or "W n")
2714 elseif objecttype == "stop_clip" then
2715   stop_pdf_code()
2716   miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2717 elseif objecttype == "special" then

```

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

```

2718         if prescript and prescript.postmplibverbtex then
2719             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtex
2720         end
2721     elseif objecttype == "text" then
2722         local ot = object.transform -- 3,4,5,6,1,2
2723         start_pdf_code()
2724         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
2725         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
2726         stop_pdf_code()
2727     elseif not trgroup and fading_ ~= "stop" then
2728         local evenodd, collect, both = false, false, false
2729         local postscript = object.postscript
2730         if not object.istext then
2731             if postscript == "evenodd" then
2732                 evenodd = true
2733             elseif postscript == "collect" then
2734                 collect = true
2735             elseif postscript == "both" then
2736                 both = true
2737             elseif postscript == "eoboth" then
2738                 evenodd = true
2739                 both = true
2740             end
2741         end
2742         if collect then
2743             if not savedpath then
2744                 savedpath = { object.path or false }
2745                 savedhtap = { object.htap or false }
2746             else
2747                 savedpath[#savedpath+1] = object.path or false
2748                 savedhtap[#savedhtap+1] = object.htap or false
2749             end
2750         else

```

Removed from ConTeXt general: color stuff.

```

2751         local ml = object.miterlimit
2752         if ml and ml ~= miterlimit then
2753             miterlimit = ml
2754             pdf_literalcode("%f M",ml)
2755         end
2756         local lj = object.linejoin
2757         if lj and lj ~= linejoin then
2758             linejoin = lj
2759             pdf_literalcode("%i j",lj)
2760         end
2761         local lc = object.linecap
2762         if lc and lc ~= linecap then
2763             linecap = lc
2764             pdf_literalcode("%i J",lc)
2765         end
2766         local dl = object.dash
2767         if dl then
2768             local d = format("[%s] %f d",tableconcat(dl.dashes or {}," "),dl.offset)

```

```

2769         if d ~= dashed then
2770             dashed = d
2771             pdf_literalcode(dashed)
2772         end
2773     elseif dashed then
2774         pdf_literalcode("[ ] 0 d")
2775         dashed = false
2776     end
2777     local path = object.path
2778     local transformed, penwidth = false, 1
2779     local open = path and path[1].left_type and path[#path].right_type
2780     local pen = object.pen
2781     if pen then
2782         if pen.type == 'elliptical' then
2783             transformed, penwidth = pen_characteristics(object) -- boolean, value
2784             pdf_literalcode("%f w", penwidth)
2785             if objecttype == 'fill' then
2786                 objecttype = 'both'
2787             end
2788         else -- calculated by mplib itself
2789             objecttype = 'fill'
2790         end
2791     end
2792 end

```

Added : shading

```

2792     local shade_no = do_preobj_SH(object, prescript) -- shading
2793     if shade_no then
2794         pdf_literalcode("q /Pattern cs"
2795         objecttype = false
2796     end
2797     if transformed then
2798         start_pdf_code()
2799     end
2800     if path then
2801         if savedpath then
2802             for i=1, #savedpath do
2803                 local path = savedpath[i]
2804                 if transformed then
2805                     flushconcatpath(path, open)
2806                 else
2807                     flushnormalpath(path, open)
2808                 end
2809             end
2810             savedpath = nil
2811         end
2812         if transformed then
2813             flushconcatpath(path, open)
2814         else
2815             flushnormalpath(path, open)
2816         end
2817         if objecttype == "fill" then
2818             pdf_literalcode(evenodd and "h f*" or "h f")
2819         elseif objecttype == "outline" then
2820             if both then
2821                 pdf_literalcode(evenodd and "h B*" or "h B")

```

```

2822         else
2823             pdf_literalcode(open and "S" or "h S")
2824         end
2825         elseif objecttype == "both" then
2826             pdf_literalcode(evenodd and "h B*" or "h B")
2827         end
2828     end
2829     if transformed then
2830         stop_pdf_code()
2831     end
2832     local path = object.htap

```

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

```

2833     if path then
2834         if transformed then
2835             start_pdf_code()
2836         end
2837         if savedhtap then
2838             for i=1,#savedhtap do
2839                 local path = savedhtap[i]
2840                 if transformed then
2841                     flushconcatpath(path,open)
2842                 else
2843                     flushnormalpath(path,open)
2844                 end
2845             end
2846             savedhtap = nil
2847             evenodd = true
2848         end
2849         if transformed then
2850             flushconcatpath(path,open)
2851         else
2852             flushnormalpath(path,open)
2853         end
2854         if objecttype == "fill" then
2855             pdf_literalcode(evenodd and "h f*" or "h f")
2856         elseif objecttype == "outline" then
2857             pdf_literalcode(open and "S" or "h S")
2858         elseif objecttype == "both" then
2859             pdf_literalcode(evenodd and "h B*" or "h B")
2860         end
2861         if transformed then
2862             stop_pdf_code()
2863         end
2864     end

```

Added to ConTeXt general: post-object colors and shading stuff. We should beware the q ... Q scope.

```

2865         if shade_no then -- shading
2866             pdf_literalcode("W%s n /MPLibSh%s sh Q",evenodd and "*" or "",shade_no)
2867         end
2868     end
2869 end
2870 if fading_ == "start" then
2871     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}

```



```

2872         elseif trgroup == "start" then
2873             pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
2874         elseif fading_ == "stop" then
2875             local se = pdfetcs.fading.specialeffects
2876             if se then stop_special_effects(se[1], se[2], se[3]) end
2877         elseif trgroup == "stop" then
2878             local se = pdfetcs.tr_group.specialeffects
2879             if se then stop_special_effects(se[1], se[2], se[3]) end
2880         else
2881             stop_special_effects(fading_, tr_opaq, cr_over)
2882         end
2883         if fading_ or trgroup then -- extgs resetted
2884             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2885         end
2886     end
2887 end
2888 stop_pdf_code()
2889 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtex code.

```

2890     for _,v in ipairs(figcontents) do
2891         if type(v) == "table" then
2892             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
2893         else
2894             texsprint(v)
2895         end
2896     end
2897     if #figcontents.post > 0 then texsprint(figcontents.post) end
2898     figcontents = { post = { } }
2899 end
2900 end
2901 end
2902 end
2903 end
2904
2905 function luamplib.colorconverter (cr)
2906     local n = #cr
2907     if n == 4 then
2908         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
2909         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
2910     elseif n == 3 then
2911         local r, g, b = cr[1], cr[2], cr[3]
2912         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
2913     else
2914         local s = cr[1]
2915         return format("%.3f g %.3f G",s,s), "0 g 0 G"
2916     end
2917 end

```

## 2.2 T<sub>E</sub>X package

First we need to load some packages.

```

2918 \ifcsname ProvidesPackage\endcsname

```

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

```

2919 \NeedsTeXFormat{LaTeX2e}
2920 \ProvidesPackage{luamplib}
2921 [2024/12/08 v2.36.0 mplib package for LuaTeX]
2922 \fi
2923 \ifdefined\newluafunction\else
2924 \input ltluatex
2925 \fi

```

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

```

2926 \ifnum\outputmode=0
2927 \ifdefined\AddToHookNext
2928 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
2929 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
2930 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
2931 \else
2932 \input atbegshi.sty
2933 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
2934 \let\luamplibatfirstshipout\AtBeginShipoutFirst
2935 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
2936 \fi
2937 \fi

```

Loading of lua code.

```

2938 \directlua{require("luamplib")}

```

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

```

2939 \ifx\pdfoutput\undefined
2940 \let\pdfoutput\outputmode
2941 \fi
2942 \ifx\pdfliteral\undefined
2943 \protected\def\pdfliteral{\pdfextension literal}
2944 \fi

```

Set the format for `METAPOST`.

```

2945 \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.

```

2946 \ifnum\pdfoutput>0
2947 \let\mplibtoPDF\pdfliteral
2948 \else
2949 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
2950 \ifcsname PackageInfo\endcsname
2951 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
2952 \else
2953 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
2954 \fi
2955 \fi

```

To make `mplibcode` typeset always in horizontal mode.

```

2956 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
2957 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}

```

```

2958 \mplibnoforcehmode

    Catcode. We want to allow comment sign in mplibcode.
2959 \def\mplibsetupcatcodes{%
2960   %catcode`\{=12 %catcode`\}=12
2961   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_ =12
2962   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
2963 }

    Make btex...etex box zero-metric.
2964 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}

    use Transparency Group
2965 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
2966 \def\usemplibgroupmain#1{%
2967   \mplibstarttousemplibgroup
2968   \csname luamplib.group.#1\endcsname
2969   \mplibstoptousemplibgroup
2970 }
2971 \def\mplibstarttousemplibgroup{\prependtomplibbox\hbox dir TLT\bgroup}
2972 \def\mplibstoptousemplibgroup{\egroup}
2973 \protected\def\mplibgroup#1{%
2974   \begingroup
2975   \def\MP1lx{0}\def\MP1ly{0}%
2976   \def\mplibgroupname{#1}%
2977   \mplibgroupgetnexttok
2978 }
2979 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
2980 \def\mplibgroupskipsspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
2981 \def\mplibgroupbranch{%
2982   \ifx [\nexttok
2983     \expandafter\mplibgroupopts
2984   \else
2985     \ifx\mplibsptoken\nexttok
2986       \expandafter\expandafter\expandafter\mplibgroupskipsspace
2987     \else
2988       \let\mplibgroupoptions\empty
2989       \expandafter\expandafter\expandafter\mplibgroupmain
2990     \fi
2991   \fi
2992 }
2993 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
2994 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
2995 \protected\def\endmplibgroup{\egroup
2996   \directlua{ luamplib.registergroup(
2997     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
2998   )}%
2999   \endgroup
3000 }

    Patterns
3001 {\def\:{\global\let\mplibsptoken= }\ \: }
3002 \protected\def\mplibpattern#1{%
3003   \begingroup
3004   \def\mplibpatternname{#1}%
3005   \mplibpatterngetnexttok

```

```

3006 }
3007 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3008 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3009 \def\mplibpatternbranch{%
3010   \ifx [\nexttok
3011     \expandafter\mplibpatternopts
3012   \else
3013     \ifx\mplibsptoken\nexttok
3014       \expandafter\expandafter\expandafter\mplibpatternskipsspace
3015     \else
3016       \let\mplibpatternoptions\empty
3017       \expandafter\expandafter\expandafter\mplibpatternmain
3018     \fi
3019   \fi
3020 }
3021 \def\mplibpatternopts[#1]{%
3022   \def\mplibpatternoptions{#1}%
3023   \mplibpatternmain
3024 }
3025 \def\mplibpatternmain{%
3026   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3027 }
3028 \protected\def\endmpfigpattern{%
3029   \egroup
3030   \directlua{ luamplib.registerpattern(
3031     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3032   )}%
3033   \endgroup
3034 }

    simple way to use mplib: \mpfig draw fullcircle scaled 10; \endmpfig
3035 \def\mpfiginstancename{@mpfig}
3036 \protected\def\mpfig{%
3037   \begingroup
3038   \futurelet\nexttok\mplibmpfigbranch
3039 }
3040 \def\mplibmpfigbranch{%
3041   \ifx *\nexttok
3042     \expandafter\mplibprempfig
3043   \else
3044     \ifx [\nexttok
3045       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3046     \else
3047       \expandafter\expandafter\expandafter\mplibmainmpfig
3048     \fi
3049   \fi
3050 }
3051 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
3052 \def\mplibmainmpfig{%
3053   \begingroup
3054   \mplibsetupcatcodes
3055   \mplibdomainmpfig
3056 }
3057 \long\def\mplibdomainmpfig#1\endmpfig{%
3058   \endgroup

```

```

3059 \directlua{
3060   local legacy = luamplib.legacyverbatim
3061   local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3062   local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3063   luamplib.legacyverbatim = false
3064   luamplib.everymplib["\mpfiginstancename"] = ""
3065   luamplib.everyendmplib["\mpfiginstancename"] = ""
3066   luamplib.process_mplibcode(
3067     "beginfig(0) "..everympfig.." " " "[[unexpanded{#1}]] " " "..everyendmpfig.." endfig;",
3068     "\mpfiginstancename")
3069   luamplib.legacyverbatim = legacy
3070   luamplib.everymplib["\mpfiginstancename"] = everympfig
3071   luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3072 }%
3073 \endgroup
3074 }
3075 \def\mplibprempfig#1{%
3076   \begingroup
3077   \mplibsetupcatcodes
3078   \mplibdoprempfig
3079 }
3080 \long\def\mplibdoprempfig#1\endmpfig{%
3081   \endgroup
3082   \directlua{
3083     local legacy = luamplib.legacyverbatim
3084     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3085     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3086     luamplib.legacyverbatim = false
3087     luamplib.everymplib["\mpfiginstancename"] = ""
3088     luamplib.everyendmplib["\mpfiginstancename"] = ""
3089     luamplib.process_mplibcode([[unexpanded{#1}]],"\mpfiginstancename")
3090     luamplib.legacyverbatim = legacy
3091     luamplib.everymplib["\mpfiginstancename"] = everympfig
3092     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3093   }%
3094   \endgroup
3095 }
3096 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3097 \unless\ifcsname ver@luamplib.sty\endcsname
3098   \def\mplibcodegetinstancename[#1]{\gdef\currentmpinstancename{#1}\mplibcodeindeed}
3099   \protected\def\mplibcode{%
3100     \begingroup
3101     \futurelet\nexttok\mplibcodebranch
3102   }
3103   \def\mplibcodebranch{%
3104     \ifx [\nexttok
3105       \expandafter\mplibcodegetinstancename
3106     \else
3107       \global\let\currentmpinstancename\empty
3108       \expandafter\mplibcodeindeed
3109     \fi
3110   }
3111   \def\mplibcodeindeed{%

```

```

3112 \begingroup
3113 \mplibsetupcatcodes
3114 \mplibdocode
3115 }
3116 \long\def\mplibdocode#1\endmplibcode{%
3117 \endgroup
3118 \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]===],"\currentmpinstancename")}%
3119 \endgroup
3120 }
3121 \protected\def\endmplibcode{endmplibcode}
3122 \else

```

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

```

3123 \newenvironment{mplibcode}[1][{}]{%
3124 \global\def\currentmpinstancename{#1}%
3125 \mplibtmptoks{}\ltxdomplibcode
3126 }{}
3127 \def\ltxdomplibcode{%
3128 \begingroup
3129 \mplibsetupcatcodes
3130 \ltxdomplibcodeindeed
3131 }
3132 \def\mplib@mplibcode{mplibcode}
3133 \long\def\ltxdomplibcodeindeed#1\end#2{%
3134 \endgroup
3135 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3136 \def\mplibtemp@a{#2}%
3137 \ifx\mplib@mplibcode\mplibtemp@a
3138 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]===],"\currentmpinstancename")}%
3139 \end{mplibcode}%
3140 \else
3141 \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3142 \expandafter\ltxdomplibcode
3143 \fi
3144 }
3145 \fi

```

User settings.

```

3146 \def\mplibshowlog#1{\directlua{
3147 local s = string.lower("#1")
3148 if s == "enable" or s == "true" or s == "yes" then
3149 luamplib.showlog = true
3150 else
3151 luamplib.showlog = false
3152 end
3153 }}
3154 \def\mpliblegacybehavior#1{\directlua{
3155 local s = string.lower("#1")
3156 if s == "enable" or s == "true" or s == "yes" then
3157 luamplib.legacyverbatim = true
3158 else
3159 luamplib.legacyverbatim = false
3160 end
3161 }}
3162 \def\mplibverbatim#1{\directlua{

```

```

3163   local s = string.lower("#1")
3164   if s == "enable" or s == "true" or s == "yes" then
3165     luamplib.verbatiminput = true
3166   else
3167     luamplib.verbatiminput = false
3168   end
3169 }}
3170 \newtoks\mplibtmp toks
3171 \everymplib & \everyendmplib: macros resetting luamplib.every(end)mplib tables
3172 \ifcsname ver@luamplib.sty\endcsname
3173   \protected\def\everymplib{%
3174     \begingroup
3175     \mplibsetupcatcodes
3176     \mplibdoeverymplib
3177   }
3178   \protected\def\everyendmplib{%
3179     \begingroup
3180     \mplibsetupcatcodes
3181     \mplibdoeveryendmplib
3182   }
3183   \newcommand\mplibdoeverymplib[2][{}]{%
3184     \endgroup
3185     \directlua{
3186       luamplib.everymplib["#1"] = [==[\unexpanded{#2}]==]
3187     }%
3188   }
3189   \newcommand\mplibdoeveryendmplib[2][{}]{%
3190     \endgroup
3191     \directlua{
3192       luamplib.everyendmplib["#1"] = [==[\unexpanded{#2}]==]
3193     }%
3194   }
3195 \else
3196   \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3197   \protected\def\everymplib#1{%
3198     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3199     \begingroup
3200     \mplibsetupcatcodes
3201     \mplibdoeverymplib
3202   }
3203   \long\def\mplibdoeverymplib#1{%
3204     \endgroup
3205     \directlua{
3206       luamplib.everymplib["\currentmpinstancename"] = [==[\unexpanded{#1}]==]
3207     }%
3208   }
3209   \protected\def\everyendmplib#1{%
3210     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3211     \begingroup
3212     \mplibsetupcatcodes
3213     \mplibdoeveryendmplib
3214   }
3215   \long\def\mplibdoeveryendmplib#1{%

```

```

3215 \endgroup
3216 \directlua{
3217     luamplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
3218 }%
3219 }
3220 \fi

```

Allow T<sub>E</sub>X dimen/color macros. Now runscript does the job, so the following lines are not needed for most cases.

```

3221 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
3222 \def\mpcolor#1#{\domplibcolor{#1}}
3223 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

3224 \def\mplibnumbersystem#1{\directlua{
3225     local t = "#1"
3226     if t == "binary" then t = "decimal" end
3227     luamplib.numbersystem = t
3228 }}

```

Settings for .mp cache files.

```

3229 \def\mplibmakenocache#1{\mplibdomakenocache #1,*}
3230 \def\mplibdomakenocache#1,{%
3231     \ifx\empty#1\empty
3232         \expandafter\mplibdomakenocache
3233     \else
3234         \ifx*#1\else
3235             \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
3236             \expandafter\expandafter\expandafter\mplibdomakenocache
3237         \fi
3238     \fi
3239 }
3240 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*}
3241 \def\mplibdocancelnocache#1,{%
3242     \ifx\empty#1\empty
3243         \expandafter\mplibdocancelnocache
3244     \else
3245         \ifx*#1\else
3246             \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
3247             \expandafter\expandafter\expandafter\mplibdocancelnocache
3248         \fi
3249     \fi
3250 }
3251 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

3252 \def\mplibtexttextlabel#1{\directlua{
3253     local s = string.lower("#1")
3254     if s == "enable" or s == "true" or s == "yes" then
3255         luamplib.texttextlabel = true
3256     else
3257         luamplib.texttextlabel = false
3258     end
3259 }}
3260 \def\mplibcodeinherit#1{\directlua{

```



```

3261 local s = string.lower("#1")
3262 if s == "enable" or s == "true" or s == "yes" then
3263     luamplib.codeinherit = true
3264 else
3265     luamplib.codeinherit = false
3266 end
3267 }}
3268 \def\mplibglobaltexttext#1{\directlua{
3269     local s = string.lower("#1")
3270     if s == "enable" or s == "true" or s == "yes" then
3271         luamplib.globaltexttext = true
3272     else
3273         luamplib.globaltexttext = false
3274     end
3275 }}

```

The followings are from ConTeXt general, mostly.  
We use a dedicated scratchbox.

```

3276 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

3277 \def\mplibstarttoPDF#1#2#3#4{%
3278     \prependtomplibbox
3279     \hbox dir TLT\bgroup
3280     \xdef\MPllx{#1}\xdef\MPlly{#2}%
3281     \xdef\MPurx{#3}\xdef\MPury{#4}%
3282     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3283     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3284     \parskip0pt%
3285     \leftskip0pt%
3286     \parindent0pt%
3287     \everypar{}%
3288     \setbox\mplibscratchbox\vbox\bgroup
3289     \noindent
3290 }
3291 \def\mplibstoptoPDF{%
3292     \par
3293     \egroup %
3294     \setbox\mplibscratchbox\hbox %
3295     {\hskip-\MPllx bp%
3296      \raise-\MPlly bp%
3297      \box\mplibscratchbox}%
3298     \setbox\mplibscratchbox\vbox to \MPheight
3299     {\vfill
3300      \hsize\MPwidth
3301      \wd\mplibscratchbox0pt%
3302      \ht\mplibscratchbox0pt%
3303      \dp\mplibscratchbox0pt%
3304      \box\mplibscratchbox}%
3305     \wd\mplibscratchbox\MPwidth
3306     \ht\mplibscratchbox\MPheight
3307     \box\mplibscratchbox
3308     \egroup
3309 }

```

Text items have a special handler.

```

3310 \def\mplibtexttext#1#2#3#4#5{%
3311   \begingroup
3312   \setbox\mplibscratchbox\hbox
3313     {\font\temp=#1 at #2bp%
3314      \temp
3315      #3}%
3316   \setbox\mplibscratchbox\hbox
3317     {\hskip#4 bp%
3318      \raise#5 bp%
3319      \box\mplibscratchbox}%
3320   \wd\mplibscratchbox0pt%
3321   \ht\mplibscratchbox0pt%
3322   \dp\mplibscratchbox0pt%
3323   \box\mplibscratchbox
3324   \endgroup
3325 }
```

Input luamplib.cfg when it exists.

```

3326 \openin0=luamplib.cfg
3327 \ifeof0 \else
3328   \closein0
3329   \input luamplib.cfg
3330 \fi
```

Code for tagpdf

```

3331 \def\luamplibtagtextbegin#1{}
3332 \let\luamplibtagtextend\relax
3333 \let\luamplibtagasgroupbegin\relax
3334 \let\luamplibtagasgroupend\relax
3335 \ifcsname SuspendTagging\endcsname\else\endinput\fi
3336 \ifcsname ver@tagpdf.sty\endcsname \else
3337   \ExplSyntaxOn
3338   \keys_define:nn{luamplib/notag}
3339   {
3340     ,alt          .code:n = { }
3341     ,actualtext   .code:n = { }
3342     ,artifact     .code:n = { }
3343     ,text         .code:n = { }
3344     ,correct-BBox .code:n = { }
3345     ,tag          .code:n = { }
3346     ,debug        .code:n = { }
3347     ,instance     .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3348     ,instancename .meta:n = { instance = {#1} }
3349     ,unknown      .code:n = { \tl_gset:Ne \currentmpinstancename {\l_keys_key_str} }
3350   }
3351   \RenewDocumentCommand\mplibcode{0{}}
3352   {
3353     \tl_gset_eq:NN \currentmpinstancename \c_empty_tl
3354     \keys_set:nn{luamplib/notag}{#1}
3355     \mplibmptoks{}\ltxdomplibcode
3356   }
3357   \ExplSyntaxOff
3358   \let\mplibaltext \luamplibtagtextbegin
3359   \let\mplibactualtext \mplibaltext
```

```

3360 \endinput\fi
3361 \let\mplibstarttoPDForiginal\mplibstarttoPDF
3362 \let\mplibstoptoPDForiginal\mplibstoptoPDF
3363 \let\mplibputtextboxoriginal\mplibputtextbox
3364 \let\mplibstarttousemplibgrouporiginal\mplibstarttousemplibgroup
3365 \let\mplibstoptousemplibgrouporiginal\mplibstoptousemplibgroup
3366 \ExplSyntaxOn
3367 \tl_new:N \l__luamplib_tag_alt_tl
3368 \tl_new:N \l__luamplib_tag_alt_dflt_tl
3369 \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure}
3370 \tl_new:N \l__luamplib_tag_actual_tl
3371 \tl_new:N \l__luamplib_tag_struct_tl
3372 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
3373 \bool_new:N \l__luamplib_tag_usetext_bool
3374 \bool_new:N \l__luamplib_tag_BBox_bool
3375 \bool_set_true:N \l__luamplib_tag_BBox_bool
3376 \seq_new:N\l__luamplib_tag_bboxcorr_seq
3377 \bool_new:N\l__luamplib_tag_bboxcorr_bool
3378 \bool_new:N \l__luamplib_tag_debug_bool
3379 \tl_new:N \l__luamplib_BBox_label_tl
3380 \tl_new:N \l__luamplib_BBox_llx_tl
3381 \tl_new:N \l__luamplib_BBox_lly_tl
3382 \tl_new:N \l__luamplib_BBox_urx_tl
3383 \tl_new:N \l__luamplib_BBox_ury_tl
3384 \cs_set_nopar:Npn \luamplibtagtextbegin #1
3385 {
3386   \bool_if:NTF \l__luamplib_tag_usetext_bool
3387   {
3388     \tag_mc_end_push:
3389     \tag_mc_begin:n{}
3390     \tag_struct_begin:n{tag=NonStruct,stash}
3391     \def\myboxnum{#1}
3392     \edef\mystructnum{\tag_get:n{struct_num}}
3393     \edef\statebeforebox{\interval{\tag_get:n{struct_counter}+\tag_get:n{mc_counter}}}
3394   }
3395   {
3396     \tag_if_active:TF
3397     { \bool_set_true:N \l_tmpa_bool }
3398     { \bool_set_false:N \l_tmpa_bool }
3399     \SuspendTagging{luamplib.texttext}
3400   }
3401 }
3402 \cs_set_nopar:Npn \luamplibtagtextend
3403 {
3404   \bool_if:NTF \l__luamplib_tag_usetext_bool
3405   {
3406     \edef\stateafterbox{\interval{\tag_get:n{struct_counter}+\tag_get:n{mc_counter}}}
3407     \tag_if_active:T {
3408       \int_compare:nNnTF
3409       {\stateafterbox}
3410       =
3411       {\statebeforebox}
3412       { \cs_gset_nopar:cpe {luamplib.notagbox.\myboxnum} {\mystructnum} }
3413       { \cs_gset_nopar:cpe {luamplib.tagbox.\myboxnum} {\mystructnum} }

```

```

3414 }
3415 \tag_struct_end:
3416 \tag_mc_end:
3417 \tag_mc_begin_pop:n{ }
3418 }
3419 {
3420   \bool_if:NT \l_tmpa_bool
3421     { \ResumeTagging{luamplib.texttext} }
3422 }
3423 }
3424 \msg_new:nnn {luamplib}{figure-text-reuse}
3425 {
3426   texttext~box~#1~probably~is~incorrectly~tagged.\\
3427   Reusing~a~box~in~text~keyed~figures~is~strongly~discouraged.
3428 }
3429 \cs_set_nopar:Npn \mplibputtextbox #1
3430 {
3431   \vbox to 0pt{\vss\hbox to 0pt{%
3432     \bool_if:NTF \l__luamplib_tag_usetext_bool
3433       {
3434         \ResumeTagging{luamplib.puttextbox}
3435         \tag_mc_end:
3436         \cs_if_exist:cTF {luamplib.tagbox.#1}
3437         {
3438           \tag_struct_use_num:n {\csname luamplib.tagbox.#1\endcsname}
3439           \raise\dp#1\copy#1
3440         }
3441         {
3442           \cs_if_exist:cF {luamplib.notagbox.#1}
3443           {
3444             \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
3445           }
3446           \tag_mc_begin:n{ }
3447           \int_set:Nn \l_tmpa_int {#1}
3448           \tag_mc_reset_box:N \l_tmpa_int
3449           \raise\dp#1\copy#1
3450           \tag_mc_end:
3451         }
3452         \tag_mc_begin:n{artifact}
3453       }
3454       {
3455         \int_set:Nn \l_tmpa_int {#1}
3456         \tag_mc_reset_box:N \l_tmpa_int
3457         \raise\dp#1\copy#1
3458       }
3459     \hss}}
3460 }
3461 \cs_new_nopar:Npn \__luamplib_tagging_begin_figure:
3462 {
3463   \tag_if_active:T
3464   {
3465     \tag_mc_end_push:
3466     \tl_if_empty:NT\l__luamplib_tag_alt_tl
3467     {

```

```

3468     \msg_warning:nne{luamplib}{alt-text-missing}{\l__luamplib_tag_alt_dflt_tl}
3469     \tl_set:N\l__luamplib_tag_alt_tl {\l__luamplib_tag_alt_dflt_tl}
3470   }
3471   \tag_struct_begin:n
3472   {
3473     tag=\l__luamplib_tag_struct_tl,
3474     alt=\l__luamplib_tag_alt_tl,
3475   }
3476   \tag_mc_begin:n{}
3477 }
3478 }
3479 \cs_new_nopar:Npn \__luamplib_tagging_end_figure:
3480 {
3481   \tag_if_active:T
3482   {
3483     \tag_mc_end:
3484     \tag_struct_end:
3485     \tag_mc_begin_pop:n{}
3486   }
3487 }
3488 \cs_new_nopar:Npn \__luamplib_tagging_begin_actualtext:
3489 {
3490   \tag_if_active:T
3491   {
3492     \tag_mc_end_push:
3493     \tag_struct_begin:n
3494     {
3495       tag=Span,
3496       actualtext=\l__luamplib_tag_actual_tl,
3497     }
3498     \tag_mc_begin:n{}
3499   }
3500 }
3501 \cs_set_eq:NN \__luamplib_tagging_end_actualtext: \__luamplib_tagging_end_figure:
3502 \cs_new_nopar:Npn \__luamplib_tagging_begin_artifact:
3503 {
3504   \tag_if_active:T
3505   {
3506     \tag_mc_end_push:
3507     \tag_mc_begin:n{artifact}
3508   }
3509 }
3510 \cs_new_nopar:Npn \__luamplib_tagging_end_artifact:
3511 {
3512   \tag_if_active:T
3513   {
3514     \tag_mc_end:
3515     \tag_mc_begin_pop:n{}
3516   }
3517 }
3518 \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_figure:
3519 \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_figure:
3520 \keys_define:nn{luamplib/tag}
3521 {

```

```

3522 ,alt .code:n =
3523 {
3524   \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
3525 }
3526 ,actualtext .code:n =
3527 {
3528   \bool_set_false:N \l__luamplib_tag_BBox_bool
3529   \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
3530   \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_actualtext:
3531   \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_actualtext:
3532   \tag_if_active:T {\noindent}
3533 }
3534 ,artifact .code:n =
3535 {
3536   \bool_set_false:N \l__luamplib_tag_BBox_bool
3537   \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3538   \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3539 }
3540 ,text .code:n =
3541 {
3542   \bool_set_false:N \l__luamplib_tag_BBox_bool
3543   \bool_set_true:N \l__luamplib_tag_usetext_bool
3544   \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3545   \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3546   \tag_if_active:T {\noindent}
3547 }
3548 ,tag .code:n =
3549 {
3550   \str_case:nnF {#1}
3551   {
3552     {text}
3553     {
3554       \bool_set_false:N \l__luamplib_tag_BBox_bool
3555       \bool_set_true:N \l__luamplib_tag_usetext_bool
3556       \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3557       \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3558       \tag_if_active:T {\noindent}
3559     }
3560     {false}
3561     {
3562       \SuspendTagging{luamplib.tagfalse}
3563     }
3564   }
3565   {
3566     \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
3567   }
3568 }
3569 ,correct-BBox .code:n =
3570 {
3571   \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
3572   \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
3573 }
3574 ,debug .code:n =
3575 { \bool_set_true:N \l__luamplib_tag_debug_bool }

```

```

3576 ,instance .code:n =
3577 { \tl_gset:Nn \currentmpinstancename {#1} }
3578 ,instancename .meta:n = { instance = {#1} }
3579 ,unknown .code:n =
3580 { \tl_gset:Ne \currentmpinstancename {\l_keys_key_str} }
3581 }
3582 \cs_new_nopar:Npn \luamplibtaggingBBox
3583 {
3584 \bool_lazy_and:nnT
3585 {\tag_if_active_p:}
3586 {\l__luamplib_tag_BBox_bool}
3587 {
3588 \tl_set:Ne \l__luamplib_BBox_label_tl {luamplib.BBox.\tag_get:n{struct_num}}
3589 \tex_savepos:D
3590 \property_record:ee{\l__luamplib_BBox_label_tl}{xpos,ypos,abspage}
3591 \tl_set:Ne \l__luamplib_BBox_llx_tl
3592 {
3593 \dim_to_decimal_in_bp:n
3594 { \property_ref:een {\l__luamplib_BBox_label_tl}{xpos}{0}sp }
3595 }
3596 \tl_set:Ne \l__luamplib_BBox_lly_tl
3597 {
3598 \dim_to_decimal_in_bp:n
3599 { \property_ref:een {\l__luamplib_BBox_label_tl}{ypos}{0}sp - \dp\mplibscratchbox }
3600 }
3601 \tl_set:Ne \l__luamplib_BBox_urx_tl
3602 {
3603 \dim_to_decimal_in_bp:n
3604 { \l__luamplib_BBox_llx_tl bp + \wd\mplibscratchbox }
3605 }
3606 \tl_set:Ne \l__luamplib_BBox_ury_tl
3607 {
3608 \dim_to_decimal_in_bp:n
3609 { \l__luamplib_BBox_lly_tl bp + \ht\mplibscratchbox + \dp\mplibscratchbox }
3610 }
3611 \bool_if:NT \l__luamplib_tag_bboxcorr_bool
3612 {
3613 \tl_set:Ne \l__luamplib_BBox_llx_tl
3614 {
3615 \fp_eval:n
3616 {
3617 \l__luamplib_BBox_llx_tl
3618 +
3619 \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {1} }
3620 }
3621 }
3622 \tl_set:Ne \l__luamplib_BBox_lly_tl
3623 {
3624 \fp_eval:n
3625 {
3626 \l__luamplib_BBox_lly_tl
3627 +
3628 \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {2} }
3629 }

```

```

3630     }
3631     \tl_set:Nc \l__luamplib_BBox_urx_tl
3632     {
3633         \fp_eval:n
3634         {
3635             \l__luamplib_BBox_urx_tl
3636             +
3637             \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {3} }
3638         }
3639     }
3640     \tl_set:Nc \l__luamplib_BBox_ury_tl
3641     {
3642         \fp_eval:n
3643         {
3644             \l__luamplib_BBox_ury_tl
3645             +
3646             \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {4} }
3647         }
3648     }
3649 }
3650 \prop_gput:cne
3651 { g__tag_struct_\tag_get:n{struct_num}_prop }
3652 {A}
3653 {
3654     << /O /Layout /BBox [
3655         \l__luamplib_BBox_llx_tl\c_space_tl
3656         \l__luamplib_BBox_lly_tl\c_space_tl
3657         \l__luamplib_BBox_urx_tl\c_space_tl
3658         \l__luamplib_BBox_ury_tl
3659     ] >>
3660 }
3661 \bool_if:NT \l__luamplib_tag_debug_bool
3662 {
3663     \iow_log:e
3664     {
3665         luamplib/tag/debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
3666         \l__luamplib_BBox_llx_tl\c_space_tl
3667         \l__luamplib_BBox_lly_tl\c_space_tl
3668         \l__luamplib_BBox_urx_tl\c_space_tl
3669         \l__luamplib_BBox_ury_tl
3670     }
3671     \use:e
3672     {
3673         \exp_not:N\AddToHookNext{shipout/foreground}
3674         {
3675             \exp_not:N\int_compare:nNnT
3676             {\exp_not:N\g_shipout_readonly_int}
3677             =
3678             {\property_ref:een{\l__luamplib_BBox_label_tl}{abspage}{0}}
3679             {
3680                 \exp_not:N\put
3681                 (\l__luamplib_BBox_llx_tl bp, \dim_eval:n{\l__luamplib_BBox_lly_tl bp -\paperheight})
3682                 {
3683                     \exp_not:N\opacity_select:n{0.5} \exp_not:N\color_select:n{red}

```



```

3684         \exp_not:N\rule
3685         {\dim_eval:n {\l__luamplib_BBox_urx_tl bp - \l__luamplib_BBox_llx_tl bp}}
3686         {\dim_eval:n {\l__luamplib_BBox_ury_tl bp - \l__luamplib_BBox_lly_tl bp}}
3687     }
3688 }
3689 }
3690 }
3691 }
3692 }
3693 }
3694 \cs_set_nopar:Npn \luamplibtagasgroupbegin
3695 {
3696   \bool_if:NT \l__luamplib_tag_usetext_bool
3697   {
3698     \ResumeTagging{luamplib.asgroup}
3699     \tag_mc_begin:n{ }
3700   }
3701 }
3702 \cs_set_nopar:Npn \luamplibtagasgroupend
3703 {
3704   \bool_if:NT \l__luamplib_tag_usetext_bool
3705   {
3706     \tag_mc_end:
3707     \SuspendTagging{luamplib.asgroup}
3708   }
3709 }
3710 \cs_set_nopar:Npn \mplibstarttousemplibgroup
3711 {
3712   \prependtomplibbox\hbox dir TLT\bgroup
3713   \luamplibtaggingbegin
3714   \setbox\mplibscratchbox\hbox\bgroup
3715   \bool_if:NT \l__luamplib_tag_usetext_bool
3716   {
3717     \tag_mc_end:
3718     \tag_mc_begin:n{ }
3719   }
3720 }
3721 \cs_set_nopar:Npn \mplibstoptousemplibgroup
3722 {
3723   \bool_if:NT \l__luamplib_tag_usetext_bool
3724   {
3725     \tag_mc_end:
3726     \tag_mc_begin:n{artifact}
3727   }
3728   \egroup
3729   \luamplibtaggingBBox
3730   \unhbox\mplibscratchbox
3731   \luamplibtaggingend
3732   \egroup
3733 }
3734 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
3735 {
3736   \prependtomplibbox
3737   \hbox dir TLT\bgroup

```

```

3738 \luamplibtaggingbegin % begin tagging
3739 \xdef\MPllx{#1}\xdef\MPlly{#2}%
3740 \xdef\MPurx{#3}\xdef\MPury{#4}%
3741 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3742 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3743 \parskip0pt
3744 \leftskip0pt
3745 \parindent0pt
3746 \everypar{}%
3747 \setbox\mplibscratchbox\vbox\bgroup
3748 \SuspendTagging{luamplib.mplibtopdf}% stop tag inside figure
3749 \noindent
3750 }
3751 \cs_set_nopar:Npn \mplibstoptoPDF
3752 {
3753   \par
3754   \egroup
3755   \setbox\mplibscratchbox\hbox
3756     {\hskip-\MPllx bp
3757      \raise-\MPlly bp
3758      \box\mplibscratchbox}%
3759   \setbox\mplibscratchbox\vbox to \MPheight
3760     {\vfill
3761      \hsize\MPwidth
3762      \wd\mplibscratchbox0pt
3763      \ht\mplibscratchbox0pt
3764      \dp\mplibscratchbox0pt
3765      \box\mplibscratchbox}%
3766   \wd\mplibscratchbox\MPwidth
3767   \ht\mplibscratchbox\MPheight
3768   \luamplibtaggingBBox % BBox
3769   \box\mplibscratchbox
3770   \luamplibtaggingend % end tagging
3771   \egroup
3772 }
3773 \RenewDocumentCommand\mplibcode{O{}}
3774 {
3775   \msg_set:nnn {luamplib}{alt-text-missing}
3776   {
3777     Alternative~text~for~mplibcode~is~missing.\
3778     Using~the~default~value~'#1'~instead.
3779   }
3780   \tl_gset_eq:NN \currentmpinstancename \c_empty_tl
3781   \keys_set:nn{luamplib/tag}{#1}
3782   \tl_if_empty:NF \currentmpinstancename
3783   { \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure~\currentmpinstancename} }
3784   \mplibtmp toks{}\ltxdomplibcode
3785 }
3786 \RenewDocumentCommand\mpfig{s O{}}
3787 {
3788   \begingroup
3789   \IfBooleanTF{#1}
3790   {\mplibprempfig *}
3791   {

```

```

3792 \msg_set:nnn {luamplib}{alt-text-missing}
3793 {
3794   Alternative~text~for~mpfig~is~missing.\\
3795   Using~the~default~value~'##1'~instead.
3796 }
3797 \keys_set:nn{luamplib/tag}{#2}
3798 \tl_if_empty:NF \mpfiginstancename
3799 { \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure~\mpfiginstancename} }
3800 \mplibmainmpfig
3801 }
3802 }
3803 \RenewDocumentCommand\usemplibgroup{0}{ m}
3804 {
3805   \beginingroup
3806   \msg_set:nnn {luamplib}{alt-text-missing}
3807   {
3808     Alternative~text~for~usemplibgroup~is~missing.\\
3809     Using~the~default~value~'##1'~instead.
3810   }
3811   \keys_set:nn{luamplib/tag}{#1}
3812   \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure~#2}
3813   \mplibstarttousemplibgroup
3814   \csname luamplib.group.#2\endcsname
3815   \mplibstoptousemplibgroup
3816   \endgroup
3817 }
3818 \cs_new_nopar:Npn \mplibaltext #1
3819 {
3820   \tl_set:Nn \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
3821 }
3822 \cs_new_nopar:Npn \mplibactualtext #1
3823 {
3824   \tl_set:Nn \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
3825 }
3826 \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.

<div>GNU GENERAL PUBLIC LICENSE</div> <div>Version 2, June 1991</div> <div>Copyright © 1989, 1991 Free Software Foundation, Inc.</div> <div>51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA</div> <div>Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.</div> <div>    <b>Preamble</b></div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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 granted for everyone's free use or not licensed at all.</div> <div>The precise terms and conditions for copying, distribution and modification follow:</div> <div>    <b>TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION</b></div> <div>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".</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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:</div> <div>(a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.</div> <div>(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.</div> <div>(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.)</div> <div>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</div>		<div>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.</div> <div>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.</div> <div>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.</div> <div>4. You may copy and distribute the Program for 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:</div> <div>(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,</div> <div>(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,</div> <div>(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.)</div> <div>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 binary 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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.</div> <div>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.</div>		<div>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.</div> <div>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.</div> <div>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.</div> <div>    <b>NO WARRANTY</b></div> <div>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.</div> <div>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 RE-DISTRIBUTE 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.</div> <div>    <b>END OF TERMS AND CONDITIONS</b></div> <div>    <b>Appendix: How to Apply These Terms to Your New Programs</b></div> <div>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.</div> <div>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.</div> <div>    one line to give the program's name and a brief idea of what it does.     Copyright (C) yyyy name of author</div> <div>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.</div> <div>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.</div> <div>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.</div> <div>Also add information on how to contact you by electronic and paper mail.</div> <div>If the program is interactive, make it output a short notice like this when it starts in an interactive mode:</div> <div>    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.</div> <div>The hypothetical commands <code>show w</code> and <code>show c</code> should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than <code>show w</code> and <code>show c</code>; they could even be mouse-clicks or menu items—whatever suits your program.</div> <div>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:</div> <div>    Voyodnyne, Inc., hereby disclaims all copyright interest in the program     "Gnomovision" (which makes passes at compilers) written by James     Hacker.</div> <div>    signature of Ty Coon, 1 April 1989     Ty Coon, President of Vice</div> <div>This General Public License does not permit incorporating your program into proprietary programs. If your program is a subruntime 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.</div>
--	--	--	--	---