

# The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/01/27 v2.38.3

## Abstract

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

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	$\TeX$	3
1.1.1	<code>\mplibforcehmode</code>	3
1.1.2	<code>\everymplib, \everyendmplib</code>	3
1.1.3	<code>\mplibsetformat</code>	3
1.1.4	<code>\mplibnumbersystem</code>	4
1.1.5	<code>\mplibshowlog</code>	4
1.1.6	<code>\mpliblegacybehavior</code>	4
1.1.7	<code>\mplibtexttextlabel</code>	5
1.1.8	<code>\mplibcodeinherit</code>	6
1.1.9	<code>\mplibglobaltexttext</code>	6
1.1.10	Separate METAPOST instances	6
1.1.11	<code>\mplibverbatim</code>	7
1.1.12	<code>\mpdim</code>	7
1.1.13	<code>\mpcolor</code>	7
1.1.14	<code>\mpfig, \endmpfig</code>	8
1.1.15	About cache files	8
1.1.16	About figure box metric	9
1.1.17	<code>luamplib.cfg</code>	9
1.1.18	Tagged PDF	9
1.2	METAPOST	11
1.2.1	<code>mplibdimen, mplibcolor</code>	11
1.2.2	<code>mplibtexcolor, mplibrgbtexcolor</code>	11
1.2.3	<code>withmplibcolors</code>	11
1.2.4	<code>withtransparency</code>	12

1.2.5	<code>withshadingmethod</code> . . . . .	12
1.2.6	<code>withfademethod</code> . . . . .	13
1.2.7	<code>mplibgraphicstext</code> . . . . .	14
1.2.8	<code>mplibglyph</code> . . . . .	15
1.2.9	<code>mplibdrawglyph</code> , and its friends . . . . .	15
1.2.10	<code>mpliboutlinetext</code> . . . . .	16
1.2.11	<code>\mppattern</code> , with <code>mppattern</code> . . . . .	16
1.2.12	<code>asgroup</code> . . . . .	19
1.2.13	<code>\mplibgroup</code> . . . . .	20
1.2.14	<code>mpliblength</code> , <code>mplibuclength</code> . . . . .	21
1.2.15	<code>mplibsubstring</code> , <code>mplibucsubstring</code> . . . . .	21
1.3	Lua . . . . .	21
1.3.1	<code>runscript</code> . . . . .	21
1.3.2	<code>luamplib.instances</code> . . . . .	22
1.3.3	<code>luamplib.process_mplibcode</code> . . . . .	22
1.3.4	<code>luamplib.registerpattern</code> . . . . .	22
1.3.5	<code>luamplib.registergroup</code> . . . . .	23
<b>2</b>	<b>Implementation</b> . . . . .	<b>23</b>
2.1	Lua module . . . . .	23
2.2	$\TeX$ package . . . . .	91
<b>3</b>	<b>The GNU GPL License v2</b> . . . . .	<b>112</b>

## 1 Documentation

This package aims at providing a simple way to typeset directly METAPOST code in a document with Lua $\TeX$ . Lua $\TeX$  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  $\LaTeX$  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 Con $\TeX$ t. They have been adapted to  $\LaTeX$  and Plain by Elie Roux and Philipp Gesang and new functionalities have been added by Kim Dohyun. The most notable changes are:

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

- Possibility to use `verbatimtex ... etex` to run a  $\TeX$  code. `VerbatimTeX`  $\langle string \rangle$  is a more versatile macro corresponding to `verbatimtex` command. Of course the behavior cannot be the same as the stand-alone `mpost`, so that you cannot include `\documentclass`, `\usepackage` etc. When these  $\TeX$  commands are found in `verbatimtex ... etex`, the entire code will be ignored.

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

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

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

## 1.1 $\TeX$

### 1.1.1 `\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.<sup>1</sup>

### 1.1.2 `\everymplib{...}, \everyendmplib{...}`

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

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



### 1.1.3 `\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 § 1.2.10). You can try other effects as well, though we did not fully tested their proper functioning.

---

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

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

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

**shading** (texdoc metafun §8.3) One thing worth mentioning about shading is: when a color expression is given in string type, it is regarded by luamplib as a color expression of T<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 with extended functionality. See below § 1.2.5.

**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, cannot properly render the isolated or knockout effect.

Transparency group is available with *plain* format as well with extended functionality. See below § 1.2.12.

#### 1.1.4 \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}.

#### 1.1.5 \mplibshowlog{enable|disable}

Default: disable. When \mplibshowlog{enable}<sup>2</sup> is declared, log messages returned by the METAPOST process will be printed to the .log file. This is the T<sub>E</sub>X side interface for luamplib.showlog.

#### 1.1.6 \mpliblegacybehavior{enable|disable}

**Legacy behavior** By default, \mpliblegacybehavior{enable} is already declared for backward compatibility, in which case T<sub>E</sub>X 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.<sup>3</sup>

```
\mplibcode
verbatimtex \moveright 3cm etex; beginfig(0); ... endfig;
```

---

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

<sup>3</sup>But the recommended way to reuse a figure is using \mplibgroup command. See below § 1.2.13.

```

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

```


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

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

```

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

```



diameter: 22.62764bp.

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

```

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

```

ABC DEF GHI

### 1.1.7 `\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  $\TeX$ .

---

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

### 1.1.8 `\mplibcodeinherit{enable|disable}`

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

### 1.1.9 `\mplibglobaltexttext{enable|disable}`

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

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



### 1.1.10 Separate `METAPOST` instances

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

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

The behavior is as follows.

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

In parallel with this functionality, we support optional argument of instance name for `\everymplib` and `\everyendmplib`, affecting only those `mplibcode` environments of the same name.

Unnamed `\everymplib` affects not only those instances with no name, but also those with name but with no corresponding `\everymplib`. The syntax is:

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

#### 1.1.11 `\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 § 1.1.12 and § 1.1.13), all other  $\TeX$  commands outside of the `btex` or `verbatimtex ... etex` are not expanded and will be fed literally to the `mplib` library.

#### 1.1.12 `\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.

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



#### 1.1.13 `\mpcolor[...]{...}`

With `\mpcolor` command, color names or expressions of color, `xcolor` and `l3color` module/packages can be used in the `mplibcode` environment (after `withcolor` command, in principle). See the example above at § 1.1.12. The optional [...] denotes the option of `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.

N.B. Formerly, only the first object would have been colored as intended among multiple graphical objects in a `METAPOST` image, because `\mpcolor` always produced `withprescript` command internally. Since v2.38.1, now that `\mpcolor` returns a `METAPOST` color expression if possible, users can issue the sentence as follows without worrying about the location of the color command:

```
draw image (drawarrow (left--right) scaled 5)
  scaled 7
  withcolor \mpcolor{red!50}
;
```



Be aware, however, that even after v2.38.1 `\mpcolor` will still insert the `withprescript` command when the color specified is a spot color (or named color in DVI mode). Users therefore have to revise the code so that the color can have effect inside the image. For instance:

```
draw image (drawarrow (left--right) scaled 5 withcolor \mpcolor{spotA})
  scaled 7
;
```

#### 1.1.14 `\mpfig ... \endmpfig`

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

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

and the starred version is roughly the same as follows:

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

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

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

Box 1

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

#### 1.1.15 About cache files

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

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

where `⟨filename⟩` is a filename 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.

#### 1.1.16 About figure box metric

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

#### 1.1.17 luamplib.cfg

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

#### 1.1.18 Tagged PDF

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

**alt**=`<text>` starts a Figure tag by default and sets an alternate text of the figure from the `<text>`.

BBox info will be added automatically to the PDF. This key is needed for ordinary `METAPOST` figures, for which, if no `alt` text is given, a default text will be used with a warning issued. You can change the alternate text within `METAPOST` code as well: `VerbatimTeX "\mplibalttext{<text>}"`;

**actualtext**=`<text>` starts a Span tag implicitly and sets a replacement text (a.k.a. actual text) from the `<text>`. If in vertical mode, horizontal mode will be forced by `\noindent` command.<sup>5</sup>

BBox info will not be added. This key is intended for figures which can be represented by a character or a small sequence of characters. You can change the actual text within `METAPOST` code as well: `VerbatimTeX "\mplibactualtext{<text>}"`;

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

**text** starts an Artifact MC but enables tagging on  $\TeX$ -text boxes (such as `btex ... etex`, excluding pictures made by `infont` operator). If in vertical mode, horizontal mode will be

---

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

forced by `\noindent` command.<sup>6</sup> BBox info will not be added. This key is intended for figures the meaning of which is the sequence of texts in the  $\TeX$ -text boxes in the order they are drawn in the figure. Inside text-mode figures, reusing  $\TeX$ -text boxes is strongly discouraged.

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

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

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

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

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

**tagging-setup=⟨key-val list⟩** This key accepts as its value the list of key-value options mentioned so far.

You can set these options anywhere in the document by declaring `\SetKeys[luamplib/tagging]{⟨key-val list⟩}`, which will affect `mplib` figures thereafter in the scope. And the options listed above are provided for `\mpfig` and `\usemplibgroup` (see [below § 1.2.12](#)) commands as well.

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

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

<sup>6</sup>The key text also shares the limitation mentioned in the previous footnote.

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

<sup>8</sup>Beware that this bypasses  $\TeX$ 's regular math formula tagging, for which the `text` key is needed.

```

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

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

```

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

## 1.2 METAPOST

### 1.2.1 `mplibdimen ...`, `mplibcolor ...`

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

### 1.2.2 `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` command.<sup>9</sup> For instance:

```

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

```

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

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

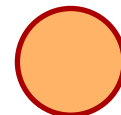
### 1.2.3 `withmplibcolors (...)`

Unlike the `withcolor` command, users can specify one color for filling and another color for stroking using the macro `withmplibcolors` at the end of a sentence. The syntax is `withmplibcolors (<fill color expr>, <stroke color expr>)`. When the argument is in string type, it is regarded as the color expression of  $\TeX$  side. A simple example (see also the example at § 1.2.9):

```

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

```



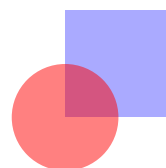
<sup>9</sup>Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color or a named color in DVI mode.

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

#### 1.2.4 withtransparency (... , ...)

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

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

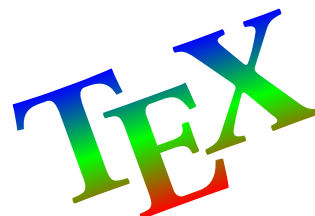


#### 1.2.5 ... withshadingmethod ...

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

- *Textual pictures* as well as paths can have shading effect. The term *textual picture* means a picture generated by `btex ... etex`, `texttext`, `TEX`, `maketext`, `mplibgraphictext` (see below § 1.2.7), or `infont` operator, though technically only the last one is a true textual picture. Note that the transparency group in which path or text objects are filled without color (see below § 1.2.12 and § 1.2.13) can also be regarded as a textual picture.

```
draw btex \bfseries\TeX etex rotated 20 scaled 6
  withshadingmethod "linear"
  withshadingvector (0,3)
  withshadingstep (
    withshadingfraction 1/2
    withshadingcolors (red,green)
  )
  withshadingstep (
    withshadingfraction 1
    withshadingcolors (green,blue)
  )
;
```



- When you give shading effect to a picture generated 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 in this case.

As shown, the syntax is  $\langle path \rangle | \langle textual\ picture \rangle$  withshadingmethod  $\langle string \rangle$ , where the latter shall be either "linear" or "circular". Other macros for optional values are:

withshadingvector  $\langle pair \rangle$  Starting and ending points (as time value) on the path.

withshadingdirection  $\langle pair \rangle$  Starting and ending points (as time value) on the bounding box.  
Default value:  $(0,2)$

withshadingorigin  $\langle pair \rangle$  The center of starting and ending circles. Default value: center p, where p is the operand of withshadingmethod.

withshadingradius  $\langle pair \rangle$  Radii of starting and ending circles. This is no-op in linear mode.  
Default value:  $(0, \text{abs}(\text{center } p - \text{urcorner } p))$

withshadingfactor  $\langle numeric \rangle$  Multiplier of the radii. This is no-op in linear mode. Default value: 1.2

withshadingcenter  $\langle pair \rangle$  Values for shifting starting center. For instance,  $(0,0)$  means that the center of starting circle is center p;  $(1,1)$  means urcorner p;  $(-1,-1)$  means llcorner p.

withshadingtransform  $\langle string \rangle$  where  $\langle string \rangle$  shall be "yes" (respect transform) or "no" (ignore transform). Default value: "no" for pictures made by infont operator; "yes" for all other cases.

withshadingdomain  $\langle pair \rangle$  Limiting values of parametric variable that varies on the axis of color gradient. Default value:  $(0,1)$

withshadingstep (...) for combined shading of more than two colors.

withshadingfraction  $\langle numeric \rangle$  Fractional number of each shading step. Only meaningful with withshadingstep.

withshadingcolors  $(\langle color\ expr \rangle, \langle color\ expr \rangle)$  Starting and ending colors, default value being (white, black). String-type argument is regarded as the color expression of T<sub>E</sub>X side.

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

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

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

`withfadecenter` is a synonym of `withfadevector`.

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

`withfadebbox` ( $\langle\text{pair}\rangle$ ,  $\langle\text{pair}\rangle$ ) sets the bounding box of the fading area, default value being (llcorner p, urcorner p). Though this option is not needed in most cases, there could be cases when users want to explicitly control the bounding box. Particularly, see the description [below](#) at § 1.2.12 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
```



### 1.2.7 mplibgraphicstext ...

`mplibgraphicstext` ( $\langle\text{string}\rangle$ ) is a METAPOST operator, the effect of which is similar to that of ConTeXt’s `graphicstext` or our own `mpliboutlinetext` (see below § 1.2.10). However the syntax is somewhat different.

```
draw mplibgraphicstext "\bfseries Funny"
  rotated 20 scaled 3
  fakebold 2.3 % fontspec option
  fillcolor "red!50" % color expression
  drawcolor 2/3 blue % or strokecolor 2/3 blue
;
```

**Funny**

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

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

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

### 1.2.9 `mplibdrawglyph ...`, `mplibstrokeglyph ...`, `mplibfillandstrokeglyph ...`

As the structure of the picture returned by `mplibglyph` is quite similar to the result of `glyph` primitive, METAPOST's `draw` command will fill the inner path of the picture with the background color. In contrast, `mplibdrawglyph` *<picture>* command fills the paths according to the nonzero winding number rule. As a result, for instance, the area surrounded by inner path of "O" will remain transparent.

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

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

<sup>11</sup>But this limitation is now lifted by the introduction of `withshadingmethod`. See above § 1.2.5.

N.B. By the way, when you want fill-and-stroke effect, issuing `filldraw` command to the last path will not always produce what you want: in such cases, you have to issue the command `draw <the last path> withpostsript "both" (or "eoboth" to apply even-odd rule)`.<sup>12</sup>

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

An example:

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



#### 1.2.10 `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!33})
  (withpen pencircle scaled .2 withcolor 2/3red)
  scaled 3
;
```



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

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

#### 1.2.11 `\mppattern{...} ... \endmppattern, ... withmppattern ...`

$\TeX$  macros `\mppattern{<name>} ... \endmppattern` define a tiling pattern cell associated with the *<name>*. METAPOST command `withmppattern`, the syntax being *<cyclic path>* | *<textual picture>* `withmppattern <string>`, will fill the given path or text with the tiling pattern cell of the *<name>* by replicating it horizontally and vertically.<sup>13</sup> As said before at § 1.2.5, the *textual picture* here means any text typeset by  $\TeX$ , mostly the result of the `btex` command (and its derivatives) or the `infont` operator.

<sup>12</sup>*metafun* provides macros `nofill`, `eofill`, `fillup`, `eofillup` etc. (see *metafun* manual § 2.11), which `luamplib` with *plain* format does not provide currently.

<sup>13</sup>`withpattern` is an operator virtually the same as `withmppattern`, but the former forces a METAPOST picture. Therefore you cannot but use `draw` command with `withpattern` operator. On the other hand, if some special command is not appended (see the example just below), *<cyclic path>* `withmppattern <string>` works as intended only with `fill` or `filldraw` command.



Table 1: options for \mppattern

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

\* in string type, numbers are separated by spaces

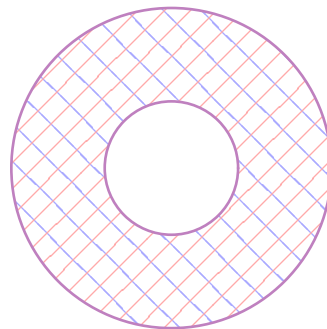
An example:

```

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

\mpfig
  draw fullcircle scaled 50
    withpostscript "collect"
  ;
  draw fullcircle scaled 120
    withmppattern "mypatt"
    withpen pencircle scaled 1
    withcolor \mpcolor{red!50!blue!50}
    withpostscript "eoboth"
  ;
\endmpfig

```



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

As for matrix option, METAPOST code such as "rotated 30 slanted .2" is allowed as well as the string or table of four numbers. You can also set xshift and yshift values by using 'shifted'

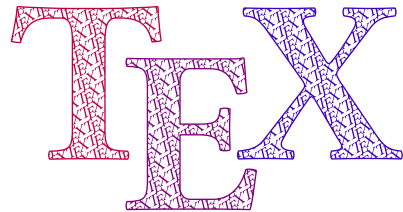
operator. But when `xshift` or `yshift` option is explicitly given, they have precedence over the effect of ‘shifted’ operator.

When you use special effect such as transparency in a pattern cell, `resources` option is needed: for instance, `resources="/ExtGState 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` (or `coloured=false`) will generate an uncolored pattern cell which shall have no color at all (i.e. `withoutcolor` command is needed for the cells made from `METAPOST` code). Uncolored pattern will be painted later by the color of a `METAPOST` object. An example:

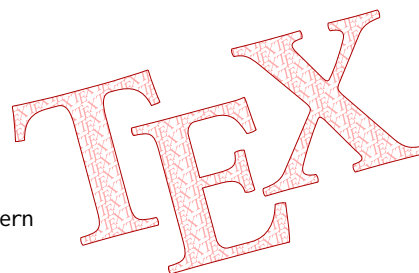
```
\begin{mppattern}{pattnocolor}
[
  colored = false,
  matrix = "slanted .3 rotated 30",
]
\tiny\TeX
\end{mppattern}

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



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

```
\begin{mplibcode}
beginfig(2)
draw mplibgraphictext "bfseries\TeX"
  fakebold 1/2
  rotated 15 scaled 8
  withmppattern "pattnocolor"
  withmplibcolors (
    2/3[red,white],      % paints the pattern
    2/3 red
  )
;
endfig;
\end{mplibcode}
```



### 1.2.12 ... asgroup ...

As said [before](#), transparency group is available with *plain* as well as *metafun* format. The syntax is exactly the same:  $\langle picture \rangle | \langle path \rangle$  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  $\langle string \rangle$`  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{ $\langle name \rangle$ }` 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 bounding box will be shifted to the origin.

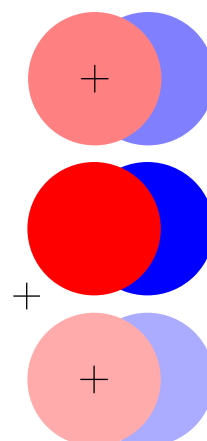
`usemplibgroup  $\langle string \rangle$`  is a METAPOST command which will add a transparency group of the name to the currentpicture. Contrary to the T<sub>E</sub>X command just mentioned, the position of the group is the same as the original transparency group.

`withgroupbbox ( $\langle pair \rangle$ ,  $\langle pair \rangle$ )` 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 50 shifted 20right withcolor blue;
  fill fullcircle scaled 50 withcolor red ;
)
asgroup ""
withgroupname "mygroup"
withtransparency (1, 1/2)
;
draw (left--right) scaled 5;
draw (up--down) scaled 5;
\endmpfig

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



```

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

```

Also note that normally the 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.

### 1.2.13 `\mplibgroup{...} ... \endmplibgroup`

These  $\text{\TeX}$  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  $\text{\TeX}$  side. The syntax is similar to the `\mppattern` command (see above § 1.2.11).

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 20 rotated 45 ;
  draw (left--right) scaled 20 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, you can reuse the `mplibgroup` using the  $\text{\TeX}$  command `\usemplibgroup` or the `METAPOST` command `usemplibgroup`. The behavior of these commands is the same as that described above at § 1.2.12, excepting that the `mplibgroup` made by  $\text{\TeX}$  code (not by `METAPOST` code) respects original height and depth.

Table 2: options for \mplibgroup

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

\* in string type, numbers are separated by spaces

#### 1.2.14 mpliblength ..., mplibuclength ...

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

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

#### 1.2.15 mplibsubstring ... of ..., mplibucsubstring ... of ...

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

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

## 1.3 Lua

### 1.3.1 runscript ...

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

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

### 1.3.2 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 `LuaTeX` manual § 11.2.8.4 (`texdoc luatex`). The following example will print `false`, `3.0`, `MetaPost` and the knots and the cyclicity of the path `unitsquare`.

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

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

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

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

3.0

### 1.3.3 Lua function `luamplib.process_mplibcode`

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

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

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

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

### 1.3.4 Lua function `luamplib.registerpattern`

This is the Lua interface for `\mppattern ... \endmppattern` described above at § 1.2.11.

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

Table 3: elements in luamplib table (partial)

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

The first argument is the register of a box containing a pattern cell, which should be prepared in advance by the user. For instance, `\setbox0=\hbox{\tiny\TeX}`, or corresponding Lua code using `tex.setbox` function; then the argument should be `0`.

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

### 1.3.5 Lua function `luamplib.registergroup`

This is the Lua interface for `\mplibgroup ... \endmplibgroup` described above at § 1.2.13.

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

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

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

Reusing an `mplibgroup`, `\usemplibgroup{<name>}`, is basically the same as running the T<sub>E</sub>X macro ‘`luamplib.group.<name>`’. If you need the `boxresource` index, inspect this macro using `token.get_macro` function.

## 2 Implementation

### 2.1 Lua module

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.38.3",

```

```

5  date          = "2026/01/27",
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

```



Provide a few “shortcuts” expected by the code.

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

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

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

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

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

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

129 local noneedtoreplace = {
130   ["boxes.mp"] = true, -- ["format.mp"] = true,
131   ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,
132   ["mpost.mp"] = true, ["plain.mp"] = true, ["rboxes.mp"] = true,
133   ["sarith.mp"] = true, ["string.mp"] = true, -- ["TEX.mp"] = true,
134   ["metafun.mp"] = true, ["metafun.mpiv"] = true, ["mp-abck.mpiv"] = true,
135   ["mp-apos.mpiv"] = true, ["mp-asnc.mpiv"] = true, ["mp-bare.mpiv"] = true,
136   ["mp-base.mpiv"] = true, ["mp-blob.mpiv"] = true, ["mp-butt.mpiv"] = true,
137   ["mp-char.mpiv"] = true, ["mp-chem.mpiv"] = true, ["mp-core.mpiv"] = true,
138   ["mp-crop.mpiv"] = true, ["mp-figs.mpiv"] = true, ["mp-form.mpiv"] = true,
139   ["mp-func.mpiv"] = true, ["mp-grap.mpiv"] = true, ["mp-grid.mpiv"] = true,
140   ["mp-grph.mpiv"] = true, ["mp-idea.mpiv"] = true, ["mp-luas.mpiv"] = true,

```

```

141 ["mp-mlib.mpiv"] = true, ["mp-node.mpiv"] = true, ["mp-page.mpiv"] = true,
142 ["mp-shap.mpiv"] = true, ["mp-step.mpiv"] = true, ["mp-text.mpiv"] = true,
143 ["mp-tool.mpiv"] = true, ["mp-cont.mpiv"] = true,
144 }
145 luamplib.noneedtoreplace = noneedtoreplace
146

```

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

```

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

```

Function luamplib.finder

```

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

```

format.mp is much complicated, so specially treated.

```

155   local function replaceformatmp(file,newfile,ofmodify)
156     local fh = ioopen(file,"r")
157     if not fh then return file end
158     local data = fh:read("*all"); fh:close()
159     fh = ioopen(newfile,"w")
160     if not fh then return file end
161     fh:write(
162       "let normalinfont = infont;\n",
163       "primarydef str infont name = rawtexttext(str) enddef;\n",
164       data,
165       "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
166       "vardef Fexp_(expr x) = rawtexttext("\${\"&decimal x&\"}$\") enddef;\n",
167       "let infont = normalinfont;\n"
168     ); fh:close()
169     lfstouch(newfile,currenttime,ofmodify)
170     return newfile
171   end
172   local function replaceinputmpfile (name,file)
173     local ofmodify = lfsattributes(file,"modification")
174     if not ofmodify then return file end
175     local newfile = name:gsub("%W","_")
176     newfile = format("%s/luamplib_input_%s", cachedir or outputdir, newfile)
177     if newfile and luamplibtime then
178       local nf = lfsattributes(newfile)
179       if nf and nf.mode == "file" and
180         ofmodify == nf.modification and luamplibtime < nf.access then
181         return nf.size == 0 and file or newfile
182       end
183     end
184     if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
185     local fh = ioopen(file,"r")

```

```

186     if not fh then return file end
187     local data = fh:read("*all"); fh:close()

```

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

```

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

```

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

```

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

```

```

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

```

For the main function: process

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

```

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

```

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

```

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

```

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

```

260     local log = l or t or "no-term"
261     log = log:gsub("%(Please type a command or say `end'%)", ""):gsub("\n+", "\n")
262     if result.status > 0 then
263         local first = log:match"(.-\n! .-)\n! "
264         if first then
265             termorlog("term", first)
266             termorlog("log", log, "Warning")
267         else
268             warn(log)
269         end
270         if result.status > 1 then
271             err(e or "see above messages")
272         end
273     elseif prevlog then

```

```
274     log = prevlog..log
```

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

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

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

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

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

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

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

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

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

```

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

```

`dvipdfmx` is supported, though nobody seems to use it.

```

354 local pdfmode = tex.outputmode > 0
355

```

`make_text` and some `run_script` uses Lua<sub>T</sub><sub>E</sub>X's `tex.runtoks`.

```

356 local catlatex = luatexbase.registernumber("catcodetable@latex")
357 local catat11 = luatexbase.registernumber("catcodetable@atletter")

```

`tex.scantoks` sometimes fail to read catcode properly, especially `\#`, `\&`, or `\%`. After some exper-

iment, we dropped using it. Instead, a function containing `tex.sprint` seems to work nicely.

```

358 local function run_tex_code (str, cat)
359   texruntoks(function() texsprint(cat or catlatex, str) end)
360 end

```

For conversion of sp to bp.

```

361 local factor = 65536*(7227/7200)
362

```

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.

```

363 local texboxes = { globalid = 0, localid = 4096 }
364 local process_tex_text
365 do
366   local texttext_fmt = 'image(addto currentpicture doublepath unitsquare \z
367     xscaled %f yscaled %f shifted (0,-%f) \z
368     withprescript "mplibtexboxid=%i:%f:%f")'
369   function process_tex_text (str, maketext)
370     if str then
371       if not maketext then str = str:gsub("\r.-$", "") end
372       local global = (has_instancename or luamplib.globaltexttext or luamplib.codeinherit)
373         and "\global" or ""
374       local tex_box_id
375       if global == "" then
376         tex_box_id = texboxes.localid + 1
377         texboxes.localid = tex_box_id
378       else
379         local boxid = texboxes.globalid + 1
380         texboxes.globalid = boxid
381         run_tex_code(format([[expandafter\newbox\csname luamplib.box.%s\endcsname]], boxid))
382         tex_box_id = tex.getcount'allocationnumber'
383       end
384       if str:find"^[taggingoff%]" then
385         str = str:gsub("^[taggingoff%]*s*", "")
386         run_tex_code(format("\luamplibnotagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
387           tex_box_id, global, tex_box_id, str))
388       else
389         run_tex_code(format("\luamplibtagtextboxset{%i}{%s\\setbox%i\\hbox{%s}}",
390           tex_box_id, global, tex_box_id, str))
391       end
392       local box = texgetbox(tex_box_id)
393       local wd = box.width / factor
394       local ht = box.height / factor
395       local dp = box.depth / factor
396       return texttext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
397     end
398   return ""

```



```

399 end
400 end
401

```

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.

```

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

```

A common function for color functions

```

416 local function colorsplit (res)
417   local t, tt = { }, res:gsub("[%%]", "", 2):explode()
418   local be = tt[1]:find"^%" and 1 or 2
419   for i=be, #tt do
420     if not tonumber(tt[i]) then break end
421     t[#t+1] = tt[i]
422   end
423   if #t == 0 then -- named color in DVI mode with no DocumentMetadata
424     run_tex_code{"\extractcolorspecs", tt[3], "\mplibtmpa\mplibtmpb"}
425     t = get_macro"mplibtmpb":explode",
426   end
427   return t
428 end
429 do
430   local colfmt = ccexplat and "l3color" or "xcolor"
431   local mplibcolorfmt = {
432     xcolor = tableconcat{
433       [[\begingroup\let\XC@mcolor\relax]],
434       [[\def\set@color{\global\mplibtmp toks\expandafter{\current@color}}]],
435       [[\color%s\endgroup]],
436     },
437     l3color = tableconcat{
438       [[\begingroup\def\__color_select:N#1{\expandafter\__color_select:nn#1}]],
439       [[\def\__color_backend_select:nn#1#2{\global\mplibtmp toks{#1 #2}}]],
440       [[\def\__kernel_backend_literal:e#1{\global\mplibtmp toks\expandafter{\expanded{#1}}}],
441       [[\color_select:n%s\endgroup]],
442     },
443   }

```

```

444 function process_color (str)
445   if str then
446     if not str:find("%b{") then
447       str = format("{%s}",str)
448     end
449     local myfmt = mplibcolorfmt[colfmt]
450     if colfmt == "l3color" and is_defined"color" then
451       if str:find("%b[") then
452         myfmt = mplibcolorfmt.xcolor
453       else
454         for _,v in ipairs(str:match"{(.+)}":explode"!") do
455           if not v:find("^%s*d+%s*$") then
456             local pp = get_macro(format("l__color_named_%s_prop",v))
457             if not pp or pp == "" then
458               myfmt = mplibcolorfmt.xcolor
459             break
460           end
461         end
462       end
463     end
464   end
465   run_tex_code(myfmt:format(str), ccexplat or catat11)
466   local t = texgettoks"mplibtmptoks"
467   if not pdfmode then
468     if t:find"^hsb" or not t:find"%d" then
469       t = "color push " .. t
470     elseif not t:find"pdf" then
471       t = t:gsub("%a+ (.+)", "pdf:bc [%1]")
472     end
473   end
474   return format('1 withprescript "mpliboverridecolor=%s"', t)
475 end
476 return ""
477 end
478 function process_mplibcolor(str)
479   local res = process_color(str)
480   if res:find"cs " or res:find"@pdf.obj" or res:find"color push" then return res end
481   res = colorsplit(res:match'"mpliboverridecolor=(.)"')
482   return format("(%s)", tableconcat(res, ","))
483 end
484 end
485
486 for \mpdim or mplibdimen
487 local function process_dimen (str)
488   if str then
489     str = str:gsub("{(.+)}", "%1")
490     run_tex_code(format([[ \mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
491     return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
492   end

```

```

492 return ""
493 end
494

```

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

```

495 local function process_verbatimtex_text (str)
496   if str then
497     run_tex_code(str)
498   end
499   return ""
500 end
501

```

For legacy verbatimtex process. verbatimtex ... etex before beginfig() is inserted just before the mplib box. And TeX code inside beginfig() ... endfig is inserted after the mplib box.

```

502 local tex_code_pre_mplib = {}
503 luamplib.figid = 1
504 luamplib.in_the_fig = false
505 local function process_verbatimtex_prefig (str)
506   if str then
507     tex_code_pre_mplib[luamplib.figid] = str
508   end
509   return ""
510 end
511 local function process_verbatimtex_infig (str)
512   if str then
513     return format('special "postmplibverbtex=%s";', str)
514   end
515   return ""
516 end
517

```

For *metafun* format. see issue #79.

```

518 mp = mp or {}
519 local mp = mp
520 mp.mf_path_reset = mp.mf_path_reset or function() end
521 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
522 mp.report = mp.report or info

```

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

```

523 catcodes = catcodes or {}
524 local catcodes = catcodes
525 catcodes.numbers = catcodes.numbers or {}
526 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
527 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
528 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
529 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
530 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
531 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
532 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex

```

533

Now `luamplib.runscript`

```
534 do
535   local runscript_funcs = {
536     luamplibtext    = process_tex_text,
537     luamplibcolor   = process_mplibcolor,
538     luamplibdimen    = process_dimen,
539     luamplibprefig   = process_verbatimtex_prefig,
540     luamplibinfig    = process_verbatimtex_infig,
541     luamplibverbtex  = process_verbatimtex_text,
542   }
```

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

```
543   local function mpprint(buffer,...)
544     for i=1,select("#",...) do
545       local value = select(i,...)
546       if value ~= nil then
547         local t = type(value)
548         if t == "number" then
549           buffer[#buffer+1] = format("%.16f",value)
550         elseif t == "string" then
551           buffer[#buffer+1] = value
552         elseif t == "table" then
553           buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
554         else -- boolean or whatever
555           buffer[#buffer+1] = tostring(value)
556         end
557       end
558     end
559   end
560   function luamplib.runscript (code)
561     local id, str = code:match("(.-){(.*)}")
562     if id and str then
563       local f = runscript_funcs[id]
564       if f then
565         local t = f(str)
566         if t then return t end
567       end
568     end
569     local f = loadstring(code)
570     if type(f) == "function" then
571       local buffer = {}
572       function mp.print(...)
573         mpprint(buffer,...)
574       end
575       local res = {f()}
576       buffer = tableconcat(buffer)
577       if buffer and buffer ~= "" then
578         return buffer
```

```

579     end
580     buffer = {}
581     mpprint(buffer, tableunpack(res))
582     return tableconcat(buffer)
583 end
584 return ""
585 end
586 end
587
    luamplib.maketext
588 luamplib.legacyverbatimtex = true
589 do

```

make\_text must be one liner, so comment sign is not allowed.

```

590 local function protecttexcontents (str)
591     return str:gsub("\\%", "\\0PerCent\0")
592           :gsub("%%.\n", "")
593           :gsub("%%.-$", "")
594           :gsub("%zPerCent%z", "\\%")
595           :gsub("\r.-$", "")
596           :gsub("%s+", " ")
597 end
598 function luamplib.maketext (str, what)
599     if str and str ~= "" then
600         str = protecttexcontents(str)
601         if what == 1 then
602             if not str:find("\\documentclass"..name_e) and
603                not str:find("\\begin%s*{document}") and
604                not str:find("\\documentstyle"..name_e) and
605                not str:find("\\usepackage"..name_e) then
606                 if luamplib.legacyverbatimtex then
607                     if luamplib.in_the_fig then
608                         return process_verbatimtex_infig(str)
609                     else
610                         return process_verbatimtex_prefig(str)
611                     end
612                 else
613                     return process_verbatimtex_text(str)
614                 end
615             end
616         else
617             return process_tex_text(str, true) -- bool is for 'char13'
618         end
619     end
620     return ""
621 end
622 end
623
    luamplib's METAPOST color operators

```

```

624 luamplib.gettexcolor = function (str, rgb)
625   local res = process_color(str):match'"mpliboverridecolor=(.+)"'
626   if res:find" cs " or res:find"@pdf.obj" then
627     if not rgb then
628       warn("%s is a spot color. Forced to CMYK", str)
629     end
630     run_tex_code({
631       "\\color_export:nnN{",
632       str,
633       "}{",
634       rgb and "space-sep-rgb" or "space-sep-cmyk",
635       "}"\mplib@tempa",
636     },ccexplat)
637     return get_macro"mplib@tempa":explode()
638   end
639   local t = colorsplit(res)
640   if #t == 3 or not rgb then return t end
641   if #t == 4 then
642     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]) }
643   end
644   return { t[1], t[1], t[1] }
645 end
646
647 luamplib.shadecolor = function (str)
648   local res = process_color(str):match'"mpliboverridecolor=(.+)"'
649   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}

650 run_tex_code({
651     [[\color_export:nnN{]], str, [[]{backend}\mplib@tempa]],
652     },ccexplat)
653 local name, value = get_macro'mplib@tempa':match'{{(.-)}}{(.-)}'
654 local t, obj = res:explode()
655 if pdfmode then
656     obj = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
657 else
658     obj = t[2]
659 end
660 return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
661 end
662 return colorsplit(res)
663 end
664

luamplib.fillandstrokecolor
665 do
666 local function graphicstextcolor (col, filldraw)
667     if col:find"^[%d%.:]+$" then
668         col = col:explode"."
669         for i=1,#col do
670             col[i] = format("%.3f", col[i])
671         end
672         if pdfmode then
673             local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
674             col[#col+1] = filldraw == "fill" and op or op:upper()
675             return tableconcat(col," ")
676         end
677         return format("[%s]", tableconcat(col," "))
678     end
679     col = process_color(col):match'"mpliboverridecolor=(.+)"'
680     if pdfmode then
681         local t = col:explode()
682         local b = filldraw == "fill" and 1 or #t/2+1
683         local e = b == 1 and #t/2 or #t
684         return tableconcat(t," ", b, e)
685     end
686     return format("[%s]", tableconcat(colorsplit(col)," "))
687 end
688 function luamplib.fillandstrokecolor (fill, stroke)
689     fill = graphicstextcolor(fill, "fill")
690     stroke = graphicstextcolor(stroke, "stroke")
691     local bc = pdfmode and "" or "pdf:bc "
692     return format('withprescript "mpliboverridecolor=%s%s %s"', bc, fill, stroke)

```



```

693 end
694 end
695

```

Remove trailing zeros for smaller PDF

```

696 local decimals = "%.d+"
697 local function rmzeros(str) return str:gsub("%.?0+$", "") end
698

```

common function for mplibgraphicstext and mpliboutlinetext

```

699 local function getrulemetric (box, curr, bp)
700   local running = -1073741824
701   local wd,ht,dp = curr.width, curr.height, curr.depth
702   wd = wd == running and box.width or wd
703   ht = ht == running and box.height or ht
704   dp = dp == running and box.depth or dp
705   if bp then
706     return wd/factor, ht/factor, dp/factor
707   end
708   return wd, ht, dp
709 end
710

```

luamplib's mplibgraphicstext operator

```

711 do
712   local emboldenfonts = { }
713   local function getemboldenwidth (curr, fakebold)
714     local width = emboldenfonts.width
715     if not width then
716       local f
717       local function getglyph(n)
718         while n do
719           if n.head then
720             getglyph(n.head)
721           elseif n.font and n.font > 0 then
722             f = n.font; break
723           end
724           n = node.getnext(n)
725         end
726       end
727       getglyph(curr)
728       width = font.getcopy(f or font.current()).size * fakebold / factor * 10
729       emboldenfonts.width = width
730     end
731     return width
732   end
733   local function getrulewhatsit (line, wd, ht, dp)
734     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
735     local pl
736     local fmt = "%f w %f %f %f %f re %s"

```

```

737   if pdfmode then
738     pl = node.new("whatsit","pdf_literal")
739     pl.mode = 0
740   else
741     fmt = "pdf:content " ..fmt
742     pl = node.new("whatsit","special")
743   end
744   pl.data = fmt:format(line, 0, -dp, wd, ht+dp, "B") :gsub(decimals,rmzeros)
745   local ss = node.new"glue"
746   node.setglue(ss, 0, 65536, 65536, 2, 2)
747   pl.next = ss
748   return pl
749 end

```

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

```

750 local tag_update_attrs
751 if is_defined"ver@tagpdf.sty" then
752   tag_update_attrs = function (n, curr)
753     while n do
754       n.attr = curr.attr
755       if n.head then
756         tag_update_attrs(n.head, curr)
757       end
758       n = node.getnext(n)
759     end
760   end
761 else
762   tag_update_attrs = function() end
763 end
764 local function embolden (box, curr, fakebold)
765   local head = curr
766   while curr do
767     if curr.head then
768       curr.head = embolden(curr, curr.head, fakebold)
769     elseif curr.replace then
770       curr.replace = embolden(box, curr.replace, fakebold)
771     elseif curr.leader then
772       if curr.leader.head then
773         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
774       elseif curr.leader.id == node.id"rule" then
775         local glue = node.effective_glue(curr, box)
776         local line = getemboldenwidth(curr, fakebold)
777         local wd,ht,dp = getrulemetric(box, curr.leader)
778         if box.id == node.id"hlist" then
779           wd = glue
780         else
781           ht, dp = 0, glue
782         end
783         local pl = getrulewhatsit(line, wd, ht, dp)

```

```

784     local pack = box.id == node.id"hlist" and node.hpack or node.vpack
785     local list = pack(pl, glue, "exactly")
786     tag_update_attrs(list,curr)
787     head = node.insert_after(head, curr, list)
788     head, curr = node.remove(head, curr)
789     end
790 elseif curr.id == node.id"rule" and curr.subtype == 0 then
791     local line = getemboldenwidth(curr, fakebold)
792     local wd,ht,dp = getrulemetric(box, curr)
793     if box.id == node.id"vlist" then
794         ht, dp = 0, ht+dp
795     end
796     local pl = getrulewhatsit(line, wd, ht, dp)
797     local list
798     if box.id == node.id"hlist" then
799         list = node.hpack(pl, wd, "exactly")
800     else
801         list = node.vpack(pl, ht+dp, "exactly")
802     end
803     tag_update_attrs(list,curr)
804     head = node.insert_after(head, curr, list)
805     head, curr = node.remove(head, curr)
806 elseif curr.id == node.id"glyph" and curr.font > 0 then
807     local f = curr.font
808     local key = format("%s:%s",f,fakebold)
809     local i = emboldenfonts[key]
810     if not i then
811         local ft = font.getfont(f) or font.getcopy(f)
812         if pdfmode then
813             width = ft.size * fakebold / factor * 10
814             emboldenfonts.width = width
815             ft.mode, ft.width = 2, width
816             i = font.define(ft)
817         else
818             if ft.format ~= "opentype" and ft.format ~= "truetype" then
819                 goto skip_type1
820             end
821             local name = ft.name:gsub('","'):gsub(';','$','')
822             name = format('%s;embolden=%s;',name,fakebold)
823             _, i = fonts.constructors.readanddefine(name,ft.size)
824         end
825         emboldenfonts[key] = i
826     end
827     curr.font = i
828 end
829 ::skip_type1::
830 curr = node.getnext(curr)
831 end
832 return head

```

```

833 end
834 luamplib.graphicstext = function (text, fakebold, fc, dc)
835     local fmt = process_tex_text(text):sub(1,-2)
836     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
837     emboldenfonts.width = nil
838     local box = texgetbox(id)
839     box.head = embolden(box, box.head, fakebold)
840     local colors = luamplib.fillandstrokecolor(fc, dc)
841     return format('%s %s)', fmt, colors)
842 end
843 end
844

```

### luamplib's mplibglyph operator

```

845 do
846     local function mperr (str)
847         return format("hide(errmessage %q)", str)
848     end
849     local function getangle (a,b,c)
850         local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
851         if r > 180 then
852             r = r - 360
853         elseif r < -180 then
854             r = r + 360
855         end
856         return r
857     end
858     local function turning (t)
859         local r, n = 0, #t
860         for i=1,2 do
861             tableinsert(t, t[i])
862         end
863         for i=1,n do
864             r = r + getangle(t[i], t[i+1], t[i+2])
865         end
866         return r/360
867     end
868     local function glyphimage(t, fmt)
869         local q,p,r = {{},{}}
870         for i,v in ipairs(t) do
871             local cmd = v[#v]
872             if cmd == "m" then
873                 p = {format('(%s,%s)',v[1],v[2])}
874                 r = {{x=v[1],y=v[2]}}
875             else
876                 local nt = t[i+1]
877                 local last = not nt or nt[#nt] == "m"
878                 if cmd == "l" then
879                     local pt = t[i-1]

```

```

880     local seco = pt[#pt] == "m"
881     if (last or seco) and r[1].x == v[1] and r[1].y == v[2] then
882     else
883         tableinsert(p, format('--(%s,%s)',v[1],v[2]))
884         tableinsert(r, {x=v[1],y=v[2]})
885     end
886     if last then
887         tableinsert(p, '--cycle')
888     end
889     elseif cmd == "c" then
890         tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
891         if last and r[1].x == v[5] and r[1].y == v[6] then
892             tableinsert(p, '..cycle')
893         else
894             tableinsert(p, format('..(%s,%s)',v[5],v[6]))
895             if last then
896                 tableinsert(p, '--cycle')
897             end
898             tableinsert(r, {x=v[5],y=v[6]})
899         end
900     else
901         return mperr"unknown operator"
902     end
903     if last then
904         tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
905     end
906 end
907 end
908 r = { }
909 if fmt == "opentype" then
910     for _,v in ipairs(q[1]) do
911         tableinsert(r, format('addto currentpicture contour %s;',v))
912     end
913     for _,v in ipairs(q[2]) do
914         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
915     end
916 else
917     for _,v in ipairs(q[2]) do
918         tableinsert(r, format('addto currentpicture contour %s;',v))
919     end
920     for _,v in ipairs(q[1]) do
921         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
922     end
923 end
924 return format('image(%s)', tableconcat(r))
925 end
926 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
927 function luamplib.glyph (f, c)
928     local filename, subfont, instance, kind, shapedata

```

```

929 local fid = tonumber(f) or font.id(f)
930 if fid > 0 then
931     local fontdata = font.getfont(fid) or font.getcopy(fid)
932     filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
933     instance = fontdata.specification and fontdata.specification.instance
934     filename = filename and filename:gsub("^harfloaded:", "")
935 else
936     local name
937     f = f:match"^%s*(.)%s*$"
938     name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
939     if not name then
940         name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
941     end
942     if not name then
943         name, subfont = f:match"(.+)%((%d+)%)$" -- Times.ttc(2)
944     end
945     name = name or f
946     subfont = (subfont or 0)+1
947     instance = instance and instance:lower()
948     for _, ftype in ipairs{"opentype", "truetype"} do
949         filename = kpse.find_file(name, ftype.." fonts")
950         if filename then
951             kind = ftype; break
952         end
953     end
954 end
955 if kind ~= "opentype" and kind ~= "truetype" then
956     f = fid and fid > 0 and tex.fontname(fid) or f
957     if kpse.find_file(f, "tfm") then
958         return format("glyph %s of %q", tonumber(c) or format("%q", c), f)
959     else
960         return mperr"font not found"
961     end
962 end
963 local time = lfsattributes(filename, "modification")
964 local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
965 local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
966 local newname = format("%s/%s.lua", cachedir or outputdir, h)
967 local newtime = lfsattributes(newname, "modification") or 0
968 if time == newtime then
969     shapedata = require(newname)
970 end
971 if not shapedata then
972     shapedata = fonts and fonts.handlers.otf.readers.loadshapes(filename, subfont, instance)
973     if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
974     table.tofile(newname, shapedata, "return")
975     lfstouch(newname, time, time)
976 end
977 local gid = tonumber(c)

```

```

978     if not gid then
979         local uni = utf8.codepoint(c)
980         for i,v in pairs(shapedata.glyphs) do
981             if c == v.name or uni == v.unicode then
982                 gid = i; break
983             end
984         end
985     end
986     if not gid then return mperr"cannot get GID (glyph id)" end
987     local fac = 1000 / (shapedata.units or 1000)
988     local t = shapedata.glyphs[gid].segments
989     if not t then return "image()" end
990     for i,v in ipairs(t) do
991         if type(v) == "table" then
992             for ii,vv in ipairs(v) do
993                 if type(vv) == "number" then
994                     t[i][ii] = format("%.0f", vv * fac)
995                 end
996             end
997         end
998     end
999     kind = shapedata.format or kind
1000     return glyphimage(t, kind)
1001 end
1002 end
1003

```

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

```

1004 do
1005     local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1006         unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1007     local outline_horz, outline_vert
1008     function outline_vert (res, box, curr, xshift, yshift)
1009         local b2u = box.dir == "LTL"
1010         local dy = (b2u and -box.depth or box.height)/factor
1011         local ody = dy
1012         while curr do
1013             if curr.id == node.id"rule" then
1014                 local wd, ht, dp = getrulemetric(box, curr, true)
1015                 local hd = ht + dp
1016                 if hd ~= 0 then
1017                     dy = dy + (b2u and dp or -ht)
1018                     if wd ~= 0 and curr.subtype == 0 then
1019                         res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1020                     end
1021                     dy = dy + (b2u and ht or -dp)
1022                 end
1023             elseif curr.id == node.id"glue" then
1024                 local vwidth = node.effective_glue(curr,box)/factor

```

```

1025     if curr.leader then
1026         local curr, kind = curr.leader, curr.subtype
1027         if curr.id == node.id"rule" then
1028             local wd = getrulemetric(box, curr, true)
1029             if wd ~= 0 then
1030                 local hd = vwidth
1031                 local dy = dy + (b2u and 0 or -hd)
1032                 if hd ~= 0 and curr.subtype == 0 then
1033                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1034                 end
1035             end
1036         elseif curr.head then
1037             local hd = (curr.height + curr.depth)/factor
1038             if hd <= vwidth then
1039                 local dy, n, iy = dy, 0, 0
1040                 if kind == 100 or kind == 103 then -- todo: gleaders
1041                     local ady = abs(ody - dy)
1042                     local ndy = math.ceil(ady / hd) * hd
1043                     local diff = ndy - ady
1044                     n = math.floor((vwidth-diff) / hd)
1045                     dy = dy + (b2u and diff or -diff)
1046                 else
1047                     n = math.floor(vwidth / hd)
1048                     if kind == 101 then
1049                         local side = vwidth % hd / 2
1050                         dy = dy + (b2u and side or -side)
1051                     elseif kind == 102 then
1052                         iy = vwidth % hd / (n+1)
1053                         dy = dy + (b2u and iy or -iy)
1054                     end
1055                 end
1056                 dy = dy + (b2u and curr.depth or -curr.height)/factor
1057                 hd = b2u and hd or -hd
1058                 iy = b2u and iy or -iy
1059                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1060                 for i=1,n do
1061                     res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1062                     dy = dy + hd + iy
1063                 end
1064             end
1065         end
1066     end
1067     dy = dy + (b2u and vwidth or -vwidth)
1068 elseif curr.id == node.id"kern" then
1069     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1070 elseif curr.id == node.id"vlist" then
1071     dy = dy + (b2u and curr.depth or -curr.height)/factor
1072     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1073     dy = dy + (b2u and curr.height or -curr.depth)/factor

```



```

1074     elseif curr.id == node.id"hlist" then
1075         dy = dy + (b2u and curr.depth or -curr.height)/factor
1076         res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1077         dy = dy + (b2u and curr.height or -curr.depth)/factor
1078     end
1079     curr = node.getnext(curr)
1080 end
1081 return res
1082 end
1083 function outline_horz (res, box, curr, xshift, yshift, discwd)
1084     local r2l = box.dir == "TRT"
1085     local dx = r2l and (discwd or box.width/factor) or 0
1086     local dirs = { { dir = r2l, dx = dx } }
1087     while curr do
1088         if curr.id == node.id"dir" then
1089             local sign, dir = curr.dir:match"(.)(...)"
1090             local level, newdir = curr.level, r2l
1091             if sign == "+" then
1092                 newdir = dir == "TRT"
1093                 if r2l ~= newdir then
1094                     local n = node.getnext(curr)
1095                     while n do
1096                         if n.id == node.id"dir" and n.level+1 == level then break end
1097                         n = node.getnext(n)
1098                     end
1099                     n = n or node.tail(curr)
1100                     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1101                 end
1102                 dirs[level] = { dir = r2l, dx = dx }
1103             else
1104                 local level = level + 1
1105                 newdir = dirs[level].dir
1106                 if r2l ~= newdir then
1107                     dx = dirs[level].dx
1108                 end
1109             end
1110             r2l = newdir
1111         elseif curr.char and curr.font and curr.font > 0 then
1112             local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1113             local gid = ft.characters[curr.char].index or curr.char
1114             local scale = ft.size / factor / 1000
1115             local slant = (ft.slant or 0)/1000
1116             local extend = (ft.extend or 1000)/1000
1117             local squeeze = (ft.squeeze or 1000)/1000
1118             local expand = 1 + (curr.expansion_factor or 0)/1000000
1119             local xscale, yscale = scale * extend * expand, scale * squeeze
1120             dx = dx - (r2l and curr.width/factor*expand or 0)
1121             local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1122             local xpos, ypos = dx + xshift + xoff, yshift + yoff

```

```

1123     local vertical = ""
1124     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1125         if ft.shared.features.vertical then -- luatexko
1126             vertical = "rotated 90"
1127             local data = ft.characters[curr.char] or { }
1128             if ft.hb then
1129                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1130                 local charraise = (ft.luatexko_charraise or 0)/factor
1131                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1132             else
1133                 local cmds = data.commands or { {0,0}, {0,0} }
1134                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1135                 xpos, ypos = xpos + hoff, ypos + voff
1136             end
1137         elseif curr ~= box.head then -- luatexja
1138             vertical = "rotated 90"
1139             local en = ft.parameters.quad/factor/2
1140             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1141         end
1142     end
1143     local image
1144     if ft.format == "opentype" or ft.format == "truetype" then
1145         image = luamplib.glyph(curr.font, gid)
1146     else
1147         local name, scale = ft.name, 1
1148         local vf = font.read_vf(name, ft.size)
1149         if vf and vf.characters[gid] then
1150             local cmds = vf.characters[gid].commands or { }
1151             for _,v in ipairs(cmds) do
1152                 if v[1] == "char" then
1153                     gid = v[2]
1154                 elseif v[1] == "font" and vf.fonts[v[2]] then
1155                     name = vf.fonts[v[2]].name
1156                     scale = vf.fonts[v[2]].size / ft.size
1157                 end
1158             end
1159         end
1160         image = format("glyph %s of %q scaled %f", gid, name, scale)
1161     end
1162     res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1163         #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1164     dx = dx + (r2l and 0 or curr.width/factor*expand)
1165 elseif curr.replace then
1166     local width = node.dimensions(curr.replace)/factor
1167     dx = dx - (r2l and width or 0)
1168     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1169     dx = dx + (r2l and 0 or width)
1170 elseif curr.id == node.id"rule" then
1171     local wd, ht, dp = getrulemetric(box, curr, true)

```

```

1172     if wd ~= 0 then
1173         local hd = ht + dp
1174         dx = dx - (r2l and wd or 0)
1175         if hd ~= 0 and curr.subtype == 0 then
1176             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1177         end
1178         dx = dx + (r2l and 0 or wd)
1179     end
1180 elseif curr.id == node.id"glue" then
1181     local width = node.effective_glue(curr, box)/factor
1182     dx = dx - (r2l and width or 0)
1183     if curr.leader then
1184         local curr, kind = curr.leader, curr.subtype
1185         if curr.id == node.id"rule" then
1186             local wd, ht, dp = getrulemetric(box, curr, true)
1187             local hd = ht + dp
1188             if hd ~= 0 then
1189                 wd = width
1190                 if wd ~= 0 and curr.subtype == 0 then
1191                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1192                 end
1193             end
1194         elseif curr.head then
1195             local wd = curr.width/factor
1196             if wd <= width then
1197                 local dx = r2l and dx+width or dx
1198                 local n, ix = 0, 0
1199                 if kind == 100 or kind == 103 then -- todo: gleaders
1200                     local adx = abs(dx-dirs[1].dx)
1201                     local ndx = math.ceil(adx / wd) * wd
1202                     local diff = ndx - adx
1203                     n = math.floor((width-diff) / wd)
1204                     dx = dx + (r2l and -diff-wd or diff)
1205                 else
1206                     n = math.floor(width / wd)
1207                     if kind == 101 then
1208                         local side = width % wd / 2
1209                         dx = dx + (r2l and -side-wd or side)
1210                     elseif kind == 102 then
1211                         ix = width % wd / (n+1)
1212                         dx = dx + (r2l and -ix-wd or ix)
1213                     end
1214                 end
1215                 wd = r2l and -wd or wd
1216                 ix = r2l and -ix or ix
1217                 local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1218                 for i=1,n do
1219                     res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1220                     dx = dx + wd + ix

```

```

1221         end
1222     end
1223 end
1224 end
1225     dx = dx + (r2l and 0 or width)
1226 elseif curr.id == node.id" kern" then
1227     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1228 elseif curr.id == node.id" math" then
1229     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1230 elseif curr.id == node.id" vlist" then
1231     dx = dx - (r2l and curr.width/factor or 0)
1232     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1233     dx = dx + (r2l and 0 or curr.width/factor)
1234 elseif curr.id == node.id" hlist" then
1235     dx = dx - (r2l and curr.width/factor or 0)
1236     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1237     dx = dx + (r2l and 0 or curr.width/factor)
1238 end
1239     curr = node.getnext(curr)
1240 end
1241 return res
1242 end
1243 function luamplib.outlinetext (text)
1244     local fmt = process_tex_text(text)
1245     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1246     local box = texgetbox(id)
1247     local res = outline_horz({ }, box, box.head, 0, 0)
1248     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1249     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1250 end
1251 end
1252

```

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

```

1253 function luamplib.getunicodegraphemes (s)
1254     local t = { }
1255     local graphemes = require'lua-uni-graphemes'
1256     for _, _, c in graphemes.graphemes(s) do
1257         table.insert(t, c)
1258     end
1259     return t
1260 end
1261 function luamplib.unicodesubstring (s,b,e,grph)
1262     local tt, t, step = { }
1263     if grph then
1264         t = luamplib.getunicodegraphemes(s)
1265     else
1266         t = { }
1267         for _, c in utf8.codes(s) do

```

```

1268     table.insert(t, utf8.char(c))
1269   end
1270 end
1271 if b <= e then
1272   b, step = b+1, 1
1273 else
1274   e, step = e+1, -1
1275 end
1276 for i = b, e, step do
1277   table.insert(tt, t[i])
1278 end
1279 s = table.concat(tt):gsub("'", "'&ditto'")
1280 return string.format("%s", s)
1281 end
1282

```

### METAPOST preambles

```

1283 luamplib.preambles = {
1284   preamble = [[
1285 boolean mplib ; mplib := true ;
1286 let dump = endinput ;
1287 let normalfontsize = fontsize;
1288 input %s ;
1289 ]],
1290   mplibcode = [[
1291 texscriptmode := 2;
1292 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1293 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1294 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1295 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1296 if known context_mlib:
1297   defaultfont := "cmitt10";
1298   let infont = normalinfont;
1299   let fontsize = normalfontsize;
1300   vardef thelabel@#(expr p,z) =
1301     if string p :
1302       thelabel@#(p infont defaultfont scaled defaultscale,z)
1303     else :
1304       p shifted (z + labeloffset*mfun_laboff@# -
1305         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1306         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1307     fi
1308   enddef;
1309 else:
1310   vardef texttext@# primary t = rawtexttext (t) enddef;
1311   def message expr t =
1312     if string t: runscript("mp.report[="&t&"]=]") else: errmessage "Not a string" fi
1313   enddef;
1314   def withtransparency (expr a, t) =

```

```

1315   withprescript "tr_alternative=" & if numeric a: decimal fi a
1316   withprescript "tr_transparency=" & decimal t
1317   enddef;
1318   vardef ddecimal primary p =
1319     decimal xpart p & " " & decimal ypart p
1320   enddef;
1321   vardef boundingbox primary p =
1322     if (path p) or (picture p) :
1323       llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1324     else :
1325       origin
1326     fi -- cycle
1327   enddef;
1328 fi
1329 def resolvedcolor(expr s) =
1330   runscript("return luamplib.shadecolor('& s &''')")
1331 enddef;
1332 def colordecimals primary c =
1333   if cmykcolor c:
1334     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1335     decimal yellowpart c & ":" & decimal blackpart c
1336   elseif rgbcolor c:
1337     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1338   elseif string c:
1339     if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1340   else:
1341     decimal c
1342   fi
1343 enddef;
1344 def externalfigure primary filename =
1345   draw rawtexttext("\includegraphics{"& filename &}")
1346 enddef;
1347 def TEX = texttext enddef;
1348 def mplibtexcolor primary c =
1349   runscript("return luamplib.gettexcolor('& c &''')")
1350 enddef;
1351 def mplibrgbtexcolor primary c =
1352   runscript("return luamplib.gettexcolor('& c &''','rgb')")
1353 enddef;
1354 def mplibgraphicstext primary t =
1355   begingroup;
1356   mplibgraphicstext_ (t)
1357 enddef;
1358 def mplibgraphicstext_ (expr t) text rest =
1359   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1360   fb, fc, dc, graphicstextpic, alsoordoublepath;
1361   picture graphicstextpic; graphicstextpic := nullpicture;
1362   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1363   let scale = scaled;

```

```

1364 def fakebold primary c = hide(fb:=c;) enddef;
1365 def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1366 def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1367 let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1368 def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1369 addto graphictextpic alsoordoublepath (origin--cycle) rest; graphictextpic:=nullpicture;
1370 def fakebold primary c = enddef;
1371 let fillcolor = fakebold; let drawcolor = fakebold;
1372 let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1373 image(draw runscript("return luamplib.graphictext([==["&t&"]===],"
1374   & decimal fb &","& fc &","& dc &")) rest;))
1375 endgroup;
1376 enddef;
1377 def mplibglyph expr c of f =
1378   runscript (
1379     "return luamplib.glyph('"
1380     & if numeric f: decimal fi f
1381     & "'',"
1382     & if numeric c: decimal fi c
1383     & "')"
1384   )
1385 enddef;
1386 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1387 def mplibdrawglyph expr g =
1388   luamplib_tmp_num_ := 0;
1389   for item within g:
1390     fill pathpart item
1391     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1392   endfor
1393 enddef;
1394 let mplibfillglyph = mplibdrawglyph;
1395 def mplibstrokeglyph expr g =
1396   luamplib_tmp_num_ := 0;
1397   for item within g:
1398     draw pathpart item
1399     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1400   endfor
1401 enddef;
1402 def mplibfillandstrokeglyph expr g =
1403   luamplib_tmp_num_ := 0;
1404   for item within g:
1405     draw pathpart item withpostscript
1406     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1407   endfor
1408 enddef;
1409 def withmplibcolors (expr f, s) =
1410   runscript("return luamplib.fillandstrokecolor('" &
1411     if not string f: colordecimals fi f & "''," &
1412     if not string s: colordecimals fi s & "'))")

```

```

1413 enddef;
1414 def mplib_do_outline_text_set_b (text f) (text d) text r =
1415   def mplib_do_outline_options_f = f enddef;
1416   def mplib_do_outline_options_d = d enddef;
1417   def mplib_do_outline_options_r = r enddef;
1418 enddef;
1419 def mplib_do_outline_text_set_f (text f) text r =
1420   def mplib_do_outline_options_f = f enddef;
1421   def mplib_do_outline_options_r = r enddef;
1422 enddef;
1423 def mplib_do_outline_text_set_u (text f) text r =
1424   def mplib_do_outline_options_f = f enddef;
1425 enddef;
1426 def mplib_do_outline_text_set_d (text d) text r =
1427   def mplib_do_outline_options_d = d enddef;
1428   def mplib_do_outline_options_r = r enddef;
1429 enddef;
1430 def mplib_do_outline_text_set_r (text d) (text f) text r =
1431   def mplib_do_outline_options_d = d enddef;
1432   def mplib_do_outline_options_f = f enddef;
1433   def mplib_do_outline_options_r = r enddef;
1434 enddef;
1435 def mplib_do_outline_text_set_n text r =
1436   def mplib_do_outline_options_r = r enddef;
1437 enddef;
1438 def mplib_do_outline_text_set_p = enddef;
1439 def mplib_fill_outline_text =
1440   for n=1 upto mpliboutlinenum:
1441     i:=0;
1442     for item within mpliboutlinepic[n]:
1443       i:=i+1;
1444       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1445       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostsript "collect"; fi
1446     endfor
1447   endfor
1448 enddef;
1449 def mplib_draw_outline_text =
1450   for n=1 upto mpliboutlinenum:
1451     for item within mpliboutlinepic[n]:
1452       draw pathpart item mplib_do_outline_options_d;
1453     endfor
1454   endfor
1455 enddef;
1456 def mplib_filldraw_outline_text =
1457   for n=1 upto mpliboutlinenum:
1458     i:=0;
1459     for item within mpliboutlinepic[n]:
1460       i:=i+1;
1461       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):

```



```

1462     fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1463     else:
1464         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1465     fi
1466 endfor
1467 endfor
1468 enddef;
1469 vardef mpliboutlinetext@# (expr t) text rest =
1470   save kind; string kind; kind := str @#;
1471   save i; numeric i;
1472   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1473   def mplib_do_outline_options_d = enddef;
1474   def mplib_do_outline_options_f = enddef;
1475   def mplib_do_outline_options_r = enddef;
1476   runscript("return luamplib.outlinetext[==["&t&"]==]");
1477   image ( addto currentpicture also image (
1478     if kind = "f":
1479       mplib_do_outline_text_set_f rest;
1480       mplib_fill_outline_text;
1481     elseif kind = "d":
1482       mplib_do_outline_text_set_d rest;
1483       mplib_draw_outline_text;
1484     elseif kind = "b":
1485       mplib_do_outline_text_set_b rest;
1486       mplib_fill_outline_text;
1487       mplib_draw_outline_text;
1488     elseif kind = "u":
1489       mplib_do_outline_text_set_u rest;
1490       mplib_filldraw_outline_text;
1491     elseif kind = "r":
1492       mplib_do_outline_text_set_r rest;
1493       mplib_draw_outline_text;
1494       mplib_fill_outline_text;
1495     elseif kind = "p":
1496       mplib_do_outline_text_set_p;
1497       mplib_draw_outline_text;
1498     else:
1499       mplib_do_outline_text_set_n rest;
1500       mplib_fill_outline_text;
1501     fi;
1502   ) mplib_do_outline_options_r; )
1503 enddef ;
1504 def withmppattern primary p =
1505   withprescript "mplibpattern=" & if numeric p: decimal fi p
1506 enddef;
1507 primarydef t withpattern p =
1508   image(
1509     if cycle t:
1510       fill

```

```

1511     else:
1512         draw
1513     fi
1514     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1515 enddef;
1516 vardef mplibtransformmatrix (text e) =
1517     save t; transform t;
1518     t = identity e;
1519     runscript("luamplib.transformmatrix = {"
1520     & decimal xpart t & ","
1521     & decimal ypart t & ","
1522     & decimal xpart t & ","
1523     & decimal ypart t & ","
1524     & decimal xpart t & ","
1525     & decimal ypart t & ","
1526     & "}");
1527 enddef;
1528 primarydef p withfademethod s =
1529     if picture p:
1530         image(
1531             draw p;
1532             draw center p withprescript "mplibfadestate=stop";
1533         )
1534     else:
1535         p withprescript "mplibfadestate=stop"
1536     fi
1537     withprescript "mplibfadetype=" & s
1538     withprescript "mplibfadebbox=" &
1539         decimal (xpart llcorner p -1/4) & ":" &
1540         decimal (ypart llcorner p -1/4) & ":" &
1541         decimal (xpart urcorner p +1/4) & ":" &
1542         decimal (ypart urcorner p +1/4)
1543 enddef;
1544 def withfadeopacity (expr a,b) =
1545     withprescript "mplibfadeopacity=" &
1546         decimal a & ":" &
1547         decimal b
1548 enddef;
1549 def withfadevector (expr a,b) =
1550     withprescript "mplibfadevector=" &
1551         decimal xpart a & ":" &
1552         decimal ypart a & ":" &
1553         decimal xpart b & ":" &
1554         decimal ypart b
1555 enddef;
1556 let withfadecenter = withfadevector;
1557 def withfaderadius (expr a,b) =
1558     withprescript "mplibfaderadius=" &
1559         decimal a & ":" &

```

```

1560     decimal b
1561 enddef;
1562 def withfadebbox (expr a,b) =
1563   withprescript "mplibfadebbox=" &
1564     decimal xpart a & ":" &
1565     decimal ypart a & ":" &
1566     decimal xpart b & ":" &
1567     decimal ypart b
1568 enddef;
1569 primarydef p asgroup s =
1570   image(
1571     draw center p
1572       withprescript "mplibgroupbbox=" &
1573         decimal (xpart llcorner p -1/4) & ":" &
1574         decimal (ypart llcorner p -1/4) & ":" &
1575         decimal (xpart urcorner p +1/4) & ":" &
1576         decimal (ypart urcorner p +1/4)
1577       withprescript "gr_state=start"
1578       withprescript "gr_type=" & s;
1579     draw p;
1580     draw center p withprescript "gr_state=stop";
1581   )
1582 enddef;
1583 def withgroupbbox (expr a,b) =
1584   withprescript "mplibgroupbbox=" &
1585     decimal xpart a & ":" &
1586     decimal ypart a & ":" &
1587     decimal xpart b & ":" &
1588     decimal ypart b
1589 enddef;
1590 def withgroupname expr s =
1591   withprescript "mplibgroupname=" & s
1592 enddef;
1593 def usemplibgroup primary s =
1594   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s &"\endcsname}")
1595   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1596 enddef;
1597 path    mplib_shade_path ;
1598 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1599 numeric mplib_shade_fx, mplib_shade_fy ;
1600 numeric mplib_shade_lx, mplib_shade_ly ;
1601 numeric mplib_shade_nx, mplib_shade_ny ;
1602 numeric mplib_shade_dx, mplib_shade_dy ;
1603 numeric mplib_shade_tx, mplib_shade_ty ;
1604 primarydef p withshadingmethod m =
1605   p
1606   if picture p :
1607     withprescript "sh_operand_type=picture"
1608     if textual p:

```

```

1609     withprescript "sh_transform=no"
1610     mplib_with_shade_method (boundingbox p, m)
1611   else:
1612     withprescript "sh_transform=yes"
1613     withprescript "sh_length_p=" & decimal length pathpart p
1614     mplib_with_shade_method (pathpart p, m)
1615   fi
1616 else :
1617   withprescript "sh_transform=yes"
1618   mplib_with_shade_method (p, m)
1619 fi
1620 enddef;
1621 def mplib_with_shade_method (expr p, m) =
1622   hide(mplib_with_shade_method_analyze(p))
1623   withprescript "sh_domain=0 1"
1624   withprescript "sh_color=into"
1625   withprescript "sh_color_a=" & colordecimals white
1626   withprescript "sh_color_b=" & colordecimals black
1627   withprescript "sh_first=" & ddecimal point 0 of p
1628   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1629   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1630   if m = "linear" :
1631     withprescript "sh_type=linear"
1632     withprescript "sh_factor=1"
1633     withprescript "sh_center_a=" & ddecimal llcorner p
1634     withprescript "sh_center_b=" & ddecimal urcorner p
1635   else :
1636     withprescript "sh_type=circular"
1637     withprescript "sh_factor=1.2"
1638     withprescript "sh_center_a=" & ddecimal center p
1639     withprescript "sh_center_b=" & ddecimal center p
1640     withprescript "sh_radius_a=" & decimal 0
1641     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1642   fi
1643 enddef;
1644 def mplib_with_shade_method_analyze(expr p) =
1645   mplib_shade_path := p ;
1646   mplib_shade_step := 1 ;
1647   mplib_shade_fx   := xpart point 0 of p ;
1648   mplib_shade_fy   := ypart point 0 of p ;
1649   mplib_shade_lx   := mplib_shade_fx ;
1650   mplib_shade_ly   := mplib_shade_fy ;
1651   mplib_shade_nx   := 0 ;
1652   mplib_shade_ny   := 0 ;
1653   mplib_shade_dx   := abs(mplib_shade_fx - mplib_shade_lx) ;
1654   mplib_shade_dy   := abs(mplib_shade_fy - mplib_shade_ly) ;
1655   for i=1 upto length(p) :
1656     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1657     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;

```

```

1658   if mplib_shade_tx > mplib_shade_dx :
1659       mplib_shade_nx := i + 1 ;
1660       mplib_shade_lx := xpart point i of p ;
1661       mplib_shade_dx := mplib_shade_tx ;
1662   fi ;
1663   if mplib_shade_ty > mplib_shade_dy :
1664       mplib_shade_ny := i + 1 ;
1665       mplib_shade_ly := ypart point i of p ;
1666       mplib_shade_dy := mplib_shade_ty ;
1667   fi ;
1668   endfor ;
1669   enddef;
1670   vardef mplib_max_radius(expr p) =
1671       max (
1672           (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1673           (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1674           (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1675           (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1676       )
1677   enddef;
1678   def withshadingstep (text t) =
1679       hide(mplib_shade_step := mplib_shade_step + 1 ;)
1680       withprescript "sh_step=" & decimal mplib_shade_step
1681       t
1682   enddef;
1683   def withshadingradius expr a =
1684       withprescript "sh_radius_a=" & decimal (xpart a)
1685       withprescript "sh_radius_b=" & decimal (ypart a)
1686   enddef;
1687   def withshadingorigin expr a =
1688       withprescript "sh_center_a=" & ddecimal a
1689       withprescript "sh_center_b=" & ddecimal a
1690   enddef;
1691   def withshadingvector expr a =
1692       withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1693       withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1694   enddef;
1695   def withshadingdirection expr a =
1696       withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1697       withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1698   enddef;
1699   def withshadingtransform expr a =
1700       withprescript "sh_transform=" & a
1701   enddef;
1702   def withshadingcenter expr a =
1703       withprescript "sh_center_a=" & ddecimal (
1704           center mplib_shade_path shifted (
1705               xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1706               ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2

```

```

1707 )
1708 )
1709 enddef;
1710 def withshadingdomain expr d =
1711   withprescript "sh_domain=" & ddecimal d
1712 enddef;
1713 def withshadingfactor expr f =
1714   withprescript "sh_factor=" & decimal f
1715 enddef;
1716 def withshadingfraction expr a =
1717   if mplib_shade_step > 0 :
1718     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1719   fi
1720 enddef;
1721 def withshadingcolors (expr a, b) =
1722   if mplib_shade_step > 0 :
1723     withprescript "sh_color=into"
1724     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1725     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1726   else :
1727     withprescript "sh_color=into"
1728     withprescript "sh_color_a=" & colordecimals a
1729     withprescript "sh_color_b=" & colordecimals b
1730   fi
1731 enddef;
1732 def withshadingstroke expr a =
1733   withprescript "sh_stroking=" & a
1734 enddef;
1735 def mpliblength primary t =
1736   runscript("return utf8.len[==[" & t & "]==]")
1737 enddef;
1738 def mplibsubstring expr p of t =
1739   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1740     & decimal xpart p & ",",
1741     & decimal ypart p & ")")
1742 enddef;
1743 def mlibuclength primary t =
1744   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
1745 enddef;
1746 def mlibucsubstring expr p of t =
1747   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1748     & decimal xpart p & ",",
1749     & decimal ypart p & ",true)")
1750 enddef;
1751 ]],
1752 legacyverbatimtex = [[
1753 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1754 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1755 let VerbatimTeX = specialVerbatimTeX;

```

```

1756 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1757 "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1758 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1759 "runscript(" &ditto&
1760 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1761 "luamplib.in_the_fig=false" &ditto& ");";
1762 ]],
1763 texttextlabel = [[
1764 let luampliboriginalinfont = infont;
1765 primarydef s infont f =
1766   if (s < char 32)
1767     or (s = char 35) % #
1768     or (s = char 36) % $
1769     or (s = char 37) % %
1770     or (s = char 38) % &
1771     or (s = char 92) % \
1772     or (s = char 94) % ^
1773     or (s = char 95) % _
1774     or (s = char 123) % {
1775     or (s = char 125) % }
1776     or (s = char 126) % ~
1777     or (s = char 127) :
1778     s luampliboriginalinfont f
1779   else :
1780     rawtexttext(s)
1781   fi
1782 enddef;
1783 def fontsize expr f =
1784   begingroup
1785     save size; numeric size;
1786     size := mplibdimen("1em");
1787     if size = 0: 10pt else: size fi
1788   endgroup
1789 enddef;
1790 ]],
1791 }
1792

```

process\_mplibcode

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

```

1793 luamplib.verbatiminput = false
1794 luamplib.everymplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1795 luamplib.everyendmplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1796 function luamplib.process_mplibcode (data, instancename)
1797   texboxes.localid = 4096

```

This is needed for legacy behavior

```

1798 if luamplib.legacyverbatim then
1799   luamplib.figid, tex_code_pre_mplib = 1, {}
1800 end

```

```

1801 local everymplib = luamplib.everymplib[instancename]
1802 local everyendmplib = luamplib.everyendmplib[instancename]
1803 data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
1804 :gsub("\r", "\n")

```

These five lines are needed for mplibverbatim mode.

```

1805 if luamplib.verbatiminput then
1806   data = data:gsub("\mpcolor%s+(-%b{ })", "mplibcolor(\"%1\")")
1807   :gsub("\mpdim%s+(< %b{ })", "mplibdimen(\"%1\")")
1808   :gsub("\mpdim%s+(< \\%a+)", "mplibdimen(\"%1\")")
1809   :gsub(btex_etex, "btex %1 etex ")
1810   :gsub(verbatimtex_etex, "verbatimtex %1 etex;")
1811 else

```

If not mplibverbatim, expand mplibcode data, so that users can use  $\TeX$  codes in it. It has turned out that no comment sign is allowed. However, we do not expand btex ... etex, verbatimtex ... etex, and string expressions.

```

1812   local t = { } -- to store btex, verbatimtex, string
1813   data = data:gsub(btex_etex, function(str)
1814     t[#t+1] = str
1815     return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
1816   end)
1817   :gsub(verbatimtex_etex, function(str)
1818     t[#t+1] = str
1819     return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
1820   end)
1821   :gsub('\"(.-)\"', function(str)
1822     t[#t+1] = str
1823     return format('\"\\unexpanded{!l!u!a!%s!m!p!l!}\"', #t)
1824   end)
1825   :gsub("\\%", "\0PerCent\0")
1826   :gsub("%%.-\n", "\n")
1827   :gsub("%zPerCent%z", "\\%")
1828   run_tex_code(format("\mplibtmptoks\\expandafter{\\expanded{%s}}",data))
1829   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

1830   :gsub("##", "#")
1831   :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
1832 end
1833 process(data, instancename)
1834 end
1835

```

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.

```

1836 local figcontents = { post = { } }
1837 local function put2output(a,...)
1838   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
1839 end

```



```

1840 local function pdf_startfigure(n,llx,lly,urx,ury)
1841   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
1842 end
1843 local function pdf_stopfigure()
1844   put2output("\mplibstoptoPDF")
1845 end

```

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

```

1846 local function pdf_literalcode (...)
1847   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
1848 end
1849 local start_pdf_code = pdfmode
1850   and function() pdf_literalcode"q" end
1851   or function() put2output"\special{pdf:bcontent}" end
1852 local stop_pdf_code = pdfmode
1853   and function() pdf_literalcode"Q" end
1854   or function() put2output"\special{pdf:econtent}" end
1855

```

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

```

1856 local function put_tex_boxes (object,prescript)
1857   local box = prescript.mplibtexboxid:explode":"
1858   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
1859   if n and tw and th then
1860     local op = object.path
1861     local first, second, fourth = op[1], op[2], op[4]
1862     local tx, ty = first.x_coord, first.y_coord
1863     local sx, rx, ry, sy = 1, 0, 0, 1
1864     if tw ~= 0 then
1865       sx = (second.x_coord - tx)/tw
1866       rx = (second.y_coord - ty)/tw
1867       if sx == 0 then sx = 0.00001 end
1868     end
1869     if th ~= 0 then
1870       sy = (fourth.y_coord - ty)/th
1871       ry = (fourth.x_coord - tx)/th
1872       if sy == 0 then sy = 0.00001 end
1873     end
1874     start_pdf_code()
1875     pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
1876     put2output("\mplibputtextbox{%i}",n)
1877     stop_pdf_code()
1878   end
1879 end
1880

```

Colors

```

1881 local do_preobj_CR
1882 do
1883   local prev_override_color

```

```

1884 function do_preobj_CR(object,prescript)
1885   if object.postscript == "collect" then return end
1886   local override = prescript and prescript.mpliboverridecolor
1887   if override then
1888     if pdfmode then
1889       pdf_literalcode(override)
1890       override = nil
1891     else
1892       put2output("\\special{%s}",override)
1893       prev_override_color = override
1894     end
1895   else
1896     local cs = object.color
1897     if cs and #cs > 0 then
1898       pdf_literalcode(luamplib.colorconverter(cs))
1899       prev_override_color = nil
1900     elseif not pdfmode then
1901       override = prev_override_color
1902       if override then
1903         put2output("\\special{%s}",override)
1904       end
1905     end
1906   end
1907   return override
1908 end
1909 end
1910

```

For transparency, shading, fading, and pattern

```

1911 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
1912 local pdfobjs, pdfetcs = {}, {}
1913 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
1914 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
1915 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
1916 local function update_pdfobjs (os, stream)
1917   local key = os
1918   if stream then key = key..stream end
1919   local on = key and pdfobjs[key]
1920   if on then
1921     return on,false
1922   end
1923   if pdfmode then
1924     if stream then
1925       on = pdf.immediateobj("stream",stream,os)
1926     elseif os then
1927       on = pdf.immediateobj(os)
1928     else
1929       on = pdf.reserveobj()
1930     end

```

```

1931 else
1932   on = pdfetcs.cnt or 1
1933   if stream then
1934     texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
1935   elseif os then
1936     texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
1937   else
1938     texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
1939   end
1940   pdfetcs.cnt = on + 1
1941 end
1942 if key then
1943   pdfobjs[key] = on
1944 end
1945 return on,true
1946 end
1947 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
1948 if pdfmode then
1949   pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
1950   local getpagers = pdfetcs.getpagers
1951   local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
1952   local initialize_resources = function (name)
1953     local tabname = format("%s_res",name)
1954     pdfetcs[tabname] = { }
1955     if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
1956       local obj = pdf.reserveobj()
1957       setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
1958       luatexbase.add_to_callback("finish_pdffile", function()
1959         pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
1960       end,
1961       format("luamplib.%s.finish_pdffile",name))
1962     end
1963   end
1964   pdfetcs.fallback_update_resources = function (name, res)
1965     local tabname = format("%s_res",name)
1966     if not pdfetcs[tabname] then
1967       initialize_resources(name)
1968     end
1969     if luatexbase.callbacktypes.finish_pdffile then
1970       local t = pdfetcs[tabname]
1971       t[#t+1] = res
1972     else
1973       local tpr, n = getpagers() or "", 0
1974       tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
1975       if n == 0 then
1976         tpr = format("%s/%s<<%s>>", tpr, name, res)
1977       end
1978       setpagers(tpr)
1979     end

```

```

1980 end
1981 else
1982   texsprint {
1983     "\\luamplibatfirstshipout{",
1984     "\\special{pdf:obj @MPLibTr<<>>}",
1985     "\\special{pdf:obj @MPLibSh<<>>}",
1986     "\\special{pdf:obj @MPLibCS<<>>}",
1987     "\\special{pdf:obj @MPLibPt<<>>}}",
1988   }
1989   pdfetcs.resadded = { }
1990   pdfetcs.fallback_update_resources = function (name,res,obj)
1991     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
1992     if not pdfetcs.resadded[name] then
1993       texsprint{"\\luampliateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
1994       pdfetcs.resadded[name] = obj
1995     end
1996   end
1997 end
1998

```

## Transparency

```

1999 local function add_extgs_resources (on, new)
2000   local key = format("MPLibTr%s", on)
2001   if new then
2002     local val = format(pdfetcs.resfmt, on)
2003     if pdfmanagement then
2004       texsprint {
2005         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2006       }
2007     else
2008       local tr = format("/%s %s", key, val)
2009       if is_defined(pdfetcs.pgfextgs) then
2010         texsprint { "\\csname ", pdfetcs.pgfextgs, "\\endcsname{" , tr, "}" }
2011       elseif is_defined"TRP@list" then
2012         texsprint(catat11,{
2013           [[\if@files\immediate\write\@auxout{]],
2014           [[\string\g@addto@macro\string\TRP@list{]],
2015           tr,
2016           [[}]\fi]],
2017         })
2018         if not get_macro"TRP@list":find(tr) then
2019           texsprint(catat11,[[\global\TRP@reruntrue]])
2020         end
2021       else
2022         pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2023       end
2024     end
2025   end
2026   return key

```

```

2027 end
2028
2029 local do_preobj_TR
2030 do
2031   local transparency_modes = {
2032     [0] = "Normal",
2033     "Normal",    "Multiply",    "Screen",    "Overlay",
2034     "SoftLight", "HardLight",   "ColorDodge", "ColorBurn",
2035     "Darken",    "Lighten",     "Difference", "Exclusion",
2036     "Hue",       "Saturation",   "Color",     "Luminosity",
2037     "Compatible",
2038     normal      = "Normal",    multiply    = "Multiply",   screen     = "Screen",
2039     overlay     = "Overlay",    softlight  = "SoftLight",  hardlight  = "HardLight",
2040     colordodge  = "ColorDodge", colorburn   = "ColorBurn",  darken     = "Darken",
2041     lighten     = "Lighten",    difference  = "Difference", exclusion   = "Exclusion",
2042     hue         = "Hue",        saturation  = "Saturation", color       = "Color",
2043     luminosity  = "Luminosity", compatible  = "Compatible",
2044   }
2045   function do_preobj_TR(object,prescript)
2046     if object.postscript == "collect" then return end
2047     local opaq = prescript and prescript.tr_transparency
2048     if opaq then
2049       local key, on, os, new
2050       local mode = prescript.tr_alternative or 1
2051       mode = transparency_modes[tonumber(mode) or mode:lower()]
2052       if not mode then
2053         mode = prescript.tr_alternative
2054         warn("unsupported blend mode: '%s'", mode)
2055       end
2056       opaq = format("%.3f", opaq) :gsub(decimals,rmzeros)
2057       for i,v in ipairs{ {mode,opaq},{ "Normal",1} } do
2058         os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[2])
2059         on, new = update_pdfobjs(os)
2060         key = add_extgs_resources(on,new)
2061         if i == 1 then
2062           pdf_literalcode("/%s gs",key)
2063         else
2064           return format("/%s gs",key)
2065         end
2066       end
2067     end
2068   end
2069 end
2070

```

Shading with *metafun* format.

```

2071 local function sh_pdfpageresources(shtype,domain,colorspace,ca,cb,coordinates,steps,fractions)
2072   for _,v in ipairs{ca,cb} do
2073     for i,vv in ipairs(v) do

```

```

2074     for ii,vvv in ipairs(vv) do
2075         v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2076     end
2077 end
2078 end
2079 local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2080 if steps > 1 then
2081     local list,bounds,encode = { },{ },{ }
2082     for i=1,steps do
2083         if i < steps then
2084             bounds[i] = format("%.3f", fractions[i] or 1)
2085         end
2086         encode[2*i-1] = 0
2087         encode[2*i] = 1
2088         os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
2089         :gsub(decimals,rmzeros)
2090         list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2091     end
2092     os = tableconcat {
2093         "<</FunctionType 3",
2094         format("/Bounds[%s]", tableconcat(bounds, ' ')),
2095         format("/Encode[%s]", tableconcat(encode, ' ')),
2096         format("/Functions[%s]", tableconcat(list, ' ')),
2097         format("/Domain[%s]>>", domain),
2098     } :gsub(decimals,rmzeros)
2099 else
2100     os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
2101     :gsub(decimals,rmzeros)
2102 end
2103 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2104 os = tableconcat {
2105     format("<</ShadingType %i", shtype),
2106     format("/ColorSpace %s", colorspace),
2107     format("/Function %s", objref),
2108     format("/Coords[%s]", coordinates),
2109     "/Extend[true true]/AntiAlias true>>",
2110 } :gsub(decimals,rmzeros)
2111 local on, new = update_pdfobjs(os)
2112 if new then
2113     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2114     if pdfmanagement then
2115         texsprint {
2116             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2117         }
2118     else
2119         local res = format("/%s %s", key, val)
2120         pdfetcs.fallback_update_resources("Shading",res,"@MPlibSh")
2121     end
2122 end

```

```

2123 return on
2124 end
2125
2126 local do_preobj_SH
2127 do
2128 pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2129   run_tex_code({
2130     [[\color_model_new:nnn]],
2131     format("{mplibcolorspace_%s}", names:gsub(",","_")),
2132     format("{DeviceN}{names={%s}}", names),
2133     [[\edef\mplib@tempa{\pdf_object_ref_last:}]],
2134   }, ccexplat)
2135   local colorspace = get_macro'mplib@tempa'
2136   t[names] = colorspace
2137   return colorspace
2138 end })
2139 local function color_normalize(ca,cb)
2140   if #cb == 1 then
2141     if #ca == 4 then
2142       cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2143     else -- #ca = 3
2144       cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2145     end
2146   elseif #cb == 3 then -- #ca == 4
2147     cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
2148   end
2149 end
2150 function do_preobj_SH(object, prescript, savedpath)
2151   local shade_no
2152   local sh_type = prescript and prescript.sh_type
2153   if not sh_type then
2154     return
2155   else
2156     local domain = prescript.sh_domain or "0 1"
2157     local centera = (prescript.sh_center_a or "0 0"):explode()
2158     local centerb = (prescript.sh_center_b or "0 0"):explode()
2159     local transform = prescript.sh_transform == "yes"
2160     local sx,sy,sr,dx,dy = 1,1,1,0,0
2161     if transform then
2162       local first = (prescript.sh_first or "0 0"):explode()
2163       local setx = (prescript.sh_set_x or "0 0"):explode()
2164       local sety = (prescript.sh_set_y or "0 0"):explode()
2165       local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2166       if x ~= 0 and y ~= 0 then
2167         local path = object.path
2168         -- avoid error upon pictures composed of multiple paths
2169         if savedpath and savedpath[1] and prescript.sh_length_p then
2170           local length_p = tonumber(prescript.sh_length_p)
2171           if length_p ~= #path then

```

```

2172         for _,v in ipairs(savedpath) do
2173             if length_p == #v then path = v; break end
2174         end
2175     end
2176 end
2177 local path1x = path[1].x_coord
2178 local path1y = path[1].y_coord
2179 local path2x = path[x].x_coord
2180 local path2y = path[y].y_coord
2181 local dxa = path2x - path1x
2182 local dya = path2y - path1y
2183 local dxb = setx[2] - first[1]
2184 local dyb = sety[2] - first[2]
2185 if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2186     sx = dxa / dxb ; if sx < 0 then sx = - sx end
2187     sy = dya / dyb ; if sy < 0 then sy = - sy end
2188     sr = math.sqrt(sx^2 + sy^2)
2189     dx = path1x - sx*first[1]
2190     dy = path1y - sy*first[2]
2191 end
2192 end
2193 end
2194 local ca, cb, colorspace, steps, fractions
2195 ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode":" }
2196 cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode":" }
2197 steps = tonumber(prescript.sh_step) or 1
2198 if steps > 1 then
2199     fractions = { prescript.sh_fraction_1 or 0 }
2200     for i=2,steps do
2201         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2202         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode":"
2203         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode":"
2204     end
2205 end
2206 if prescript.mplib_spotcolor then
2207     ca, cb = { }, { }
2208     local names, pos, objref = { }, -1, ""
2209     local script = object.prescript:explode"\13+"
2210     for i=#script,1,-1 do
2211         if script[i]:find"mplib_spotcolor" then
2212             local t, name, value = script[i]:explode"="[2]:explode":"
2213             value, objref, name = t[1], t[2], t[3]
2214             if not names[name] then
2215                 pos = pos+1
2216                 names[name] = pos
2217                 names[#names+1] = name
2218             end
2219             t = { }
2220             for j=1,names[name] do t[#t+1] = 0 end

```



```

2221         t[#t+1] = value
2222         tableinsert(#ca == #cb and ca or cb, t)
2223     end
2224 end
2225 for _,t in ipairs{ca,cb} do
2226     for _,tt in ipairs(t) do
2227         for i=1,#names-#tt do tt[#tt+1] = 0 end
2228     end
2229 end
2230 if #names == 1 then
2231     colorspace = objref
2232 else
2233     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2234 end
2235 else
2236     local model = 0
2237     for _,t in ipairs{ca,cb} do
2238         for _,tt in ipairs(t) do
2239             model = model > #tt and model or #tt
2240         end
2241     end
2242     for _,t in ipairs{ca,cb} do
2243         for _,tt in ipairs(t) do
2244             if #tt < model then
2245                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2246             end
2247         end
2248     end
2249     colorspace = model == 4 and "/DeviceCMYK"
2250                 or model == 3 and "/DeviceRGB"
2251                 or model == 1 and "/DeviceGray"
2252                 or err"unknown color model"
2253 end
2254 if sh_type == "linear" then
2255     local coordinates = format("%f %f %f %f",
2256         dx + sx*centera[1], dy + sy*centera[2],
2257         dx + sx*centerb[1], dy + sy*centerb[2])
2258     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions)
2259 elseif sh_type == "circular" then
2260     local factor = prescript.sh_factor or 1
2261     local radiusa = factor * prescript.sh_radius_a
2262     local radiusb = factor * prescript.sh_radius_b
2263     local coordinates = format("%f %f %f %f %f %f",
2264         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2265         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2266     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions)
2267 else
2268     err"unknown shading type"
2269 end

```

```

2270 end
2271 return shade_no, prescript.sh_stroking
2272 end
2273 end
2274

```

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

```

2275 if not pdfmode then
2276 pdfetcs.patternresources = {}
2277 end
2278 local function add_pattern_resources (key, val)
2279 if pdfmanagement then
2280 texsprint {
2281 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2282 }
2283 else
2284 local res = format("/%s %s", key, val)
2285 if is_defined(pdfetcs.pgfpattern) then
2286 texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2287 else
2288 pdfetcs.fallback_update_resources("Pattern",res,"@MPlibPt")
2289 if not pdfmode then
2290 tableinsert(pdfetcs.patternresources, res) -- for gather_resources()
2291 end
2292 end
2293 end
2294 end
2295 function luamplib.dolatelua (on, os)
2296 local h, v = pdf.getpos()
2297 h = format("%f", h/factor) :gsub(decimals,rmzeros)
2298 v = format("%f", v/factor) :gsub(decimals,rmzeros)
2299 if pdfmode then
2300 pdf.obj(on, format("<<s/Matrix[1 0 0 1 %s %s]>>", os, h, v))
2301 pdf.refobj(on)
2302 else
2303 local shift = os:explode()
2304 if tonumber(h) ~= tonumber(shift[1]) or tonumber(v) ~= tonumber(shift[2]) then
2305 warn([[Add 'withprescript "sh_matrixshift=%s %s"' to the picture shading]], h, v)
2306 end
2307 end
2308 end
2309 local function do_preobj_shading (object, prescript, savedpath)
2310 if not prescript or not prescript.sh_operand_type then return end
2311 local on = do_preobj_SH(object, prescript, savedpath)
2312 local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2313 on = update_pdfobjs()
2314 if pdfmode then
2315 put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(",on,",[["os,""]]}"} )
2316 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}

2317   if is_defined"RecordProperties" then
2318     put2output(tableconcat{
2319       "\\csname tex_savepos:D\\endcsname\\RecordProperties{luamplib/getpos/",on,"}{xpos,ypos}\\z
2320       \\special{pdf:put @mplibpdfobj",on," <<",os,"/Matrix[1 0 0 1 \\z
2321       \\csname dim_to_decimal_in_bp:n\\endcsname{\\RefProperty{luamplib/getpos/",on,"}{xpos}sp} \\z
2322       \\csname dim_to_decimal_in_bp:n\\endcsname{\\RefProperty{luamplib/getpos/",on,"}{ypos}sp}\\z
2323       ]>>}"
2324     })
2325   else
2326     local shift = prescript.sh_matrixshift or "0 0"
2327     texsprint{ "\\special{pdf:put @mplibpdfobj",on," <<",os,"/Matrix[1 0 0 1 ",shift,"]>>}" }
2328     put2output(tableconcat{ "\\latelua{ luamplib.dolatelua(",on,"[[",shift,"]] )" } })
2329   end
2330 end
2331 local key, val = format("MPlibPt%s", on), format(pdfetcs.resfmt, on)
2332 add_pattern_resources(key,val)
2333 pdf_literalcode("/Pattern cs/%s scn", key)
```

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

```
2334 prescript.sh_type = nil
2335 end
2336
```

### Tiling Patterns

```
2337 pdfetcs.patterns = { }
2338 local function gather_resources (optres)
2339   local t, do_pattern = { }, not optres
2340   local names = {"ExtGState", "ColorSpace", "Shading"}
2341   if do_pattern then
2342     names[#names+1] = "Pattern"
2343   end
2344   if pdfmode then
2345     if pdfmanagement then
2346       for _,v in ipairs(names) do
2347         if ltx.__pdf.Page.Resources[v] then
2348           t[#t+1] = format("/%s %s 0 R", v, ltx.pdf.object_id("__pdf/Page/Resources/"..v))
2349         end
2350       end
2351     else
2352       local res = pdfetcs.getpageres() or ""
2353       run_tex_code[["\mplibtmptoks\expandafter{\the\pdfvariable pageresources}]]
2354       res = res .. texgettoks'mplibtmptoks'
2355       if do_pattern then return res end
2356     end
2357   end
2358 end
```

```

2356     res = res:explode"/+"
2357     for _,v in ipairs(res) do
2358         v = v:match"^%s*(.)%s*$"
2359         if not v:find"Pattern" and not optres:find(v) then
2360             t[#t+1] = "/" .. v
2361         end
2362     end
2363 end
2364 else
2365     if pdfmanagement then
2366         for _,v in ipairs(names) do
2367             run_tex_code ({
2368                 "\\mplibtmptoks\\expanded{{" ,
2369                 "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/" , v , "}" ,
2370                 "{/" , v , " \\pdf_object_ref:n{__pdf/Page/Resources/" , v , "}}}" ,
2371             },ccexplat)
2372             t[#t+1] = texgettoks'mplibtmptoks'
2373         end
2374     elseif is_defined(pdfetcs.pgfgextgs) then
2375         run_tex_code ({
2376             "\\mplibtmptoks\\expanded{{" ,
2377             "\\ifpgf@sys@pdf@extgs@exists /ExtGState @pgfgextgs\\fi" ,
2378             "\\ifpgf@sys@pdf@colorspaces@exists /ColorSpace @pgfcolorspaces\\fi" ,
2379             do_pattern and "\\ifpgf@sys@pdf@patterns@exists /Pattern @pgfpatterns \\fi" or "" ,
2380             "}" ,
2381         }, catat11)
2382         t[#t+1] = texgettoks'mplibtmptoks'
2383         if pdfetcs.resadded.Shading then
2384             t[#t+1] = format("/Shading %s" , pdfetcs.resadded.Shading)
2385         end
2386     else
2387         for _,v in ipairs(names) do
2388             local vv = pdfetcs.resadded[v]
2389             if vv then
2390                 t[#t+1] = format("/%s %s" , v , vv)
2391             end
2392         end
2393     end
2394 end
2395 if do_pattern then return tableconcat(t) end
2396 -- get pattern resources
2397 local mytoks
2398 if pdfmanagement then
2399     run_tex_code ({
2400         "\\mplibtmptoks\\expanded{{" ,
2401         "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/Pattern}" ,
2402         "\\pdfdict_use:n{g__pdf_Core/Page/Resources/Pattern}}}" , "}" ,
2403     },ccexplat)
2404     mytoks = texgettoks'mplibtmptoks'

```

```

2405     if not pdfmode then
2406         mytoks = mytoks:gsub("\\str_convert_pdfname:n%s*{(.-)}", "%1") -- why not expanded?
2407     end
2408 elseif is_defined(pdfetcs.pgftxtgs) then
2409     if pdfmode then
2410         mytoks = get_macro"pgf@sys@pgf@resource@list@patterns"
2411     else
2412         local tt, abc = {}, get_macro"pgfutil@abc" or ""
2413         for v in abc:gmatch"@pgfpatterns%s*<<(.-)>>" do
2414             tt[#tt+1] = v
2415         end
2416         mytoks = tableconcat(tt)
2417     end
2418 else
2419     local tt = pdfmode and pdfetcs.Pattern_res or pdfetcs.patternresources
2420     mytoks = tt and tableconcat(tt)
2421 end
2422 if mytoks and mytoks ~= "" then
2423     t[#t+1] = format("/Pattern<<%s>>", mytoks)
2424 end
2425 return tableconcat(t)
2426 end
2427 function luamplib.registerpattern ( boxid, name, opts )
2428     local box = texgetbox(boxid)
2429     local wd = format("%.3f", box.width/factor)
2430     local hd = format("%.3f", (box.height+box.depth)/factor)
2431     info("w/h/d of pattern '%s': %s 0", name, format("%s %s", wd, hd):gsub(decimals, rmzeros))
2432     if opts.xstep == 0 then opts.xstep = nil end
2433     if opts.ystep == 0 then opts.ystep = nil end
2434     if opts.colored == nil then
2435         opts.colored = opts.coloured
2436         if opts.colored == nil then
2437             opts.colored = true
2438         end
2439     end
2440     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2441     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2442     if opts.matrix and opts.matrix:find"%a" then
2443         local data = format("mplibtransformmatrix(%s);", opts.matrix)
2444         process(data, "@mplibtransformmatrix")
2445         local t = luamplib.transformmatrix
2446         opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2447         opts.xshift = opts.xshift or format("%f", t[5])
2448         opts.yshift = opts.yshift or format("%f", t[6])
2449     end
2450     local attr = {
2451         "/Type/Pattern",
2452         "/PatternType 1",
2453         format("/PaintType %i", opts.colored and 1 or 2),

```

```

2454     "/TilingType 2",
2455     format("/XStep %s", opts.xstep or wd),
2456     format("/YStep %s", opts.ystep or hd),
2457     format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2458 }
2459 local optres = opts.resources or ""
2460 optres = optres .. gather_resources(optres)
2461 local patterns = pdfetcs.patterns
2462 if pdfmode then
2463     if opts.bbox then
2464         attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2465     end
2466     attr = tableconcat(attr) :gsub(decimals,rmzeros)
2467     local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2468     patterns[name] = { id = index, colored = opts.colored }
2469 else
2470     local cnt = #patterns + 1
2471     local objname = "@mplibpattern" .. cnt
2472     local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2473     texsprint {
2474         "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2475         "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2476         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2477         "\\special{pdf:bcontent}",
2478         "\\special{pdf:bxobj ", objname, " ", metric, "}",
2479         "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2480         "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2481         "\\special{pdf:put @resources <<", optres, ">>}",
2482         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2483         "\\special{pdf:econtent}}",
2484     }
2485     patterns[cnt] = objname
2486     patterns[name] = { id = cnt, colored = opts.colored }
2487 end
2488 end
2489
2490 local do_preobj_PAT
2491 do
2492     local function pattern_colorspace (cs)
2493         local on, new = update_pdfobjs(format("[Pattern %s]", cs))
2494         if new then
2495             local key, val = format("MPLibCS%i",on), format(pdfetcs.resfmt,on)
2496             if pdfmanagement then
2497                 texsprint {
2498                     "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
2499                 }
2500             else
2501                 local res = format("/%s %s", key, val)
2502                 if is_defined(pdfetcs.pgfcOLORSPACE) then

```

```

2503         texsprint { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
2504     else
2505         pdfetcs.fallback_update_resources("ColorSpace",res,"@MPLibCS")
2506     end
2507 end
2508 end
2509 return on
2510 end
2511 function do_preobj_PAT(object, prescript)
2512     local name = prescript and prescript.mplibpattern
2513     if not name then return end
2514     local patterns = pdfetcs.patterns
2515     local patt = patterns[name]
2516     local index = patt and patt.id or err("cannot get pattern object '%s'", name)
2517     local key = format("MPLibPt%s",index)
2518     if patt.colored then
2519         pdf_literalcode("/Pattern cs /%s scn", key)
2520     else
2521         local color = prescript.mpliboverridecolor
2522         if not color then
2523             local t = object.color
2524             color = t and #t>0 and luamplib.colorconverter(t)
2525         end
2526         if not color then return end
2527         local cs
2528         if color:find" cs " or color:find"@pdf.obj" then
2529             local t = color:explode()
2530             if pdfmode then
2531                 cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
2532                 color = t[3]
2533             else
2534                 cs = t[2]
2535                 color = t[3]:match"%[(.+)%"
2536             end
2537         else
2538             local t = colorsplit(color)
2539             cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
2540             color = tableconcat(t," ")
2541         end
2542         pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
2543     end
2544     if not patt.done then
2545         local val = pdfmode and format("%s 0 R",index) or patterns[index]
2546         add_pattern_resources(key,val)
2547     end
2548     patt.done = true
2549 end
2550 end
2551

```

## Fading

```

2552 pdfetcs.fading = { }
2553 local function do_preobj_FADE (object, prescript)
2554   local fd_type = prescript and prescript.mplibfadetype
2555   local fd_stop = prescript and prescript.mplibfadestate
2556   if not fd_type then
2557     return fd_stop -- returns "stop" (if picture) or nil
2558   end
2559   local bbox = prescript.mplibfadebbox:explode":""
2560   local dx, dy = -bbox[1], -bbox[2]
2561   local vec = prescript.mplibfadevector; vec = vec and vec:explode":""
2562   if not vec then
2563     if fd_type == "linear" then
2564       vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
2565     else
2566       local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
2567       vec = {centerx, centery, centerx, centery} -- center for both circles
2568     end
2569   end
2570   local coords = { vec[1]+dx, vec[2]+dy, vec[3]+dx, vec[4]+dy }
2571   if fd_type == "linear" then
2572     coords = format("%f %f %f %f", tableunpack(coords))
2573   elseif fd_type == "circular" then
2574     local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
2575     local radius = (prescript.mplibfaderadius or "0: "..math.sqrt(width^2+height^2)/2):explode":""
2576     tableinsert(coords, 3, radius[1])
2577     tableinsert(coords, radius[2])
2578     coords = format("%f %f %f %f %f %f", tableunpack(coords))
2579   else
2580     err("unknown fading method '%s'", fd_type)
2581   end
2582   fd_type = fd_type == "linear" and 2 or 3
2583   local opaq = (prescript.mplibfadeopacity or "1:0"):explode":""
2584   local on, os, new
2585   on = sh_pdfpageresources(fd_type, "0 1", "/DeviceGray", {{opaq[1]}}, {{opaq[2]}}, coords, 1)
2586   os = format("<</PatternType 2/Shading %s>>", format(pdfetcs.resfmt, on))
2587   on = update_pdfobjs(os)
2588   bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
2589   local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
2590   :gsub(decimals,rmzeros)
2591   os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
2592   on = update_pdfobjs(os)
2593   local resources = format(pdfetcs.resfmt, on)
2594   on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
2595   local attr = tableconcat{
2596     "/Subtype/Form",
2597     "/BBox[" .. bbox .. "]",
2598     "/Matrix[1 0 0 1 ", format("%f %f", -dx,-dy), "]",
2599     "/Resources ", resources,

```



```

2600    "/Group ", format(pdfetcs.resfmt, on),
2601  } :gsub(decimals,rmzeros)
2602  on = update_pdfobjs(attr, streamtext)
2603  os = "<</SMask<</S/Luminosity/G " .. format(pdfetcs.resfmt, on) .. ">>>>"
2604  on, new = update_pdfobjs(os)
2605  local key = add_extgs_resources(on,new)
2606  start_pdf_code()
2607  pdf_literalcode("/%s gs", key)
2608  if fd_stop then return "standalone" end
2609  return "start"
2610 end
2611

```

### Transparency Group

```

2612 pdfetcs.tr_group = { shifts = { } }
2613 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
2614 local function do_preobj_GRP (object, prescript)
2615   local grstate = prescript and prescript.gr_state
2616   if not grstate then return end
2617   local trgroup = pdfetcs.tr_group
2618   if grstate == "start" then
2619     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
2620     trgroup.isolated, trgroup.knockout = false, false
2621     for _,v in ipairs(prescript.gr_type:explode",+") do
2622       trgroup[v] = true
2623     end
2624     trgroup.bbox = prescript.mplibgroupbbox:explode":"
2625     put2output[["\begingroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
2626   elseif grstate == "stop" then
2627     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
2628     put2output(tableconcat{
2629       "\\egroup",
2630       format("\\wd\mplibscratchbox %fbp", urx-llx),
2631       format("\\ht\mplibscratchbox %fbp", ury-lly),
2632       "\\dp\mplibscratchbox 0pt",
2633     })
2634     local grattr = format("/Group<</S/Transparency/I %s/K %s>>",trgroup.isolated,trgroup.knockout)
2635     local res = gather_resources()
2636     local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
2637     if pdfmode then
2638       put2output(tableconcat{
2639         "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
2640         "/BBox[" .. bbox .. "], grattr, " } resources{" .. res .. "}" .. "\\mplibscratchbox",
2641         "\\luamplibtagasgroupput{" .. trgroup.name .. "}" .. "{",
2642         [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
2643         [[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]],
2644         [[\box\mplibscratchbox]],
2645         "}" .. "\\endgroup",
2646         "\\expandafter\\xdef\\csname luamplib.group.", trgroup.name, "\\endcsname{" ..

```

```

2647     "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
2648     "\\useboxresource \\the\\lastsavedboxresourceindex",
2649     "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
2650     "\\box\\mplibscratchbox}",
2651   })
2652   else
2653     trgroup.cnt = (trgroup.cnt or 0) + 1
2654     local objname = format("@mplibtrgr%s", trgroup.cnt)
2655     put2output(tableconcat{
2656       "\\special{pdf:boxobj ", objname, " bbox ", bbox, "}",
2657       "\\unhbox\\mplibscratchbox",
2658       "\\special{pdf:put @resources <<", res, ">>}",
2659       "\\special{pdf:exobj <<", grattr, ">>}",
2660       "\\luamplibtagasgroupput{",trgroup.name,"}{",
2661       "\\special{pdf:uxobj ", objname, "}",
2662       "}}\\endgroup",
2663     })
2664     token.set_macro("luamplib.group."..trgroup.name, tableconcat{
2665       "\\setbox\\mplibscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{",
2666       "\\special{pdf:uxobj ", objname, "}",
2667       "}}\\wd\\mplibscratchbox",urx-llx,"bp\\ht\\mplibscratchbox",ury-lly,"bp",
2668       "\\box\\mplibscratchbox",
2669       }, "global")
2670   end
2671   trgroup.shifts[trgroup.name] = { llx, lly }
2672 end
2673 return grstate
2674 end
2675 function luamplib.registergroup (boxid, name, opts)
2676   local box = texgetbox(boxid)
2677   local wd, ht, dp = node.getwhd(box)
2678   local res = (opts.resources or "") .. gather_resources()
2679   local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
2680   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2681   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2682   if opts.matrix and opts.matrix:find"%a" then
2683     local data = format("mplibtransformmatrix(%s);",opts.matrix)
2684     process(data,"@mplibtransformmatrix")
2685     opts.matrix = format("%f %f %f %f %f %f",tableunpack(luamplib.transformmatrix))
2686   end
2687   local grtype = 3
2688   if opts.bbox then
2689     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2690     grtype = 2
2691   end
2692   if opts.matrix then
2693     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
2694     grtype = opts.bbox and 4 or 1
2695   end

```

```

2696 if opts.asgroup then
2697   local t = { isolated = false, knockout = false }
2698   for _,v in ipairs(opts.asgroup:explode",+") do t[v] = true end
2699   attr[#attr+1] = format("/Group<</S/Transparency/I %s/K %s>>", t.isolated, t.knockout)
2700 end
2701 local trgroup = pdfetcs.tr_group
2702 trgroup.shifts[name] = { get_macro'MPlly', get_macro'MPlly' }
2703 local whd
2704 if pdfmode then
2705   attr = tableconcat(attr) :gsub(decimals,rmzeros)
2706   local index = tex.saveboxresource(boxid, attr, res, true, grtype)
2707   token.set_macro("luamplib.group"..name, tableconcat{
2708     "\\useboxresource ", index,
2709     }, "global")
2710   whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
2711 else
2712   trgroup.cnt = (trgroup.cnt or 0) + 1
2713   local objname = format("@mplibtrgr%s", trgroup.cnt)
2714   texsprint {
2715     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2716     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2717     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2718     "\\special{pdf:bcontent}",
2719     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
2720     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2721     "\\special{pdf:put @resources <<", res, ">>}",
2722     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2723     "\\special{pdf:econtent}}",
2724   }
2725   token.set_macro("luamplib.group"..name, tableconcat{
2726     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
2727     "\\wd\\mplibscratchbox ", wd, "sp",
2728     "\\ht\\mplibscratchbox ", ht, "sp",
2729     "\\dp\\mplibscratchbox ", dp, "sp",
2730     "\\box\\mplibscratchbox",
2731     }, "global")
2732   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
2733 end
2734 info("w/h/d of group '%s': %s", name, whd)
2735 end
2736
2737 do
2738   local function stop_special_effects(fade,opaque,over)
2739     if fade then -- fading
2740       stop_pdf_code()
2741     end
2742     if opaque then -- opacity

```

```

2743     pdf_literalcode(opaq)
2744 end
2745 if over then -- color
2746   if over:find"pdf:bc" then
2747     put2output"\special{pdf:ec}"
2748   else
2749     put2output"\special{color pop}"
2750   end
2751 end
2752 end
2753

```

For parsing prescript materials.

```

2754 local function script2table(s)
2755   local t = {}
2756   for _,i in ipairs(s:explode("\13+")) do
2757     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
2758     if k and v and k ~= "" and not t[k] then
2759       t[k] = v
2760     end
2761   end
2762   return t
2763 end
2764

```

Codes below to insert PDF lieterals are mostly from ConT<sub>E</sub>Xt general, with small changes when needed.

```

2765 local function pdf_textfigure(font,size,text,width,height,depth)
2766   text = text:gsub(".",function(c)
2767     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
2768   end)
2769   put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
2770 end
2771
2772 local bend_tolerance = 131/65536
2773
2774 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
2775
2776 local function pen_characteristics(object)
2777   local t = mplib.pen_info(object)
2778   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
2779   divider = sx*sy - rx*ry
2780   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
2781 end
2782
2783 local function concat(px, py) -- no tx, ty here
2784   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
2785 end
2786
2787 local function curved(ith,pth)

```

```

2788     local d = pth.left_x - ith.right_x
2789     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
2790        abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
2791         d = pth.left_y - ith.right_y
2792         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
2793            abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
2794             return false
2795         end
2796     end
2797     return true
2798 end
2799
2800 local function flushnormalpath(path,open)
2801     local pth, ith
2802     for i=1,#path do
2803         pth = path[i]
2804         if not ith then
2805             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
2806         elseif curved(ith,pth) then
2807             pdf_literalcode("%f %f %f %f %f %f c",
2808                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
2809         else
2810             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
2811         end
2812         ith = pth
2813     end
2814     if not open then
2815         local one = path[1]
2816         if curved(pth,one) then
2817             pdf_literalcode("%f %f %f %f %f %f c",
2818                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
2819         else
2820             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2821         end
2822     elseif #path == 1 then -- special case .. draw point
2823         local one = path[1]
2824         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2825     end
2826 end
2827
2828 local function flushconcatpath(path,open)
2829     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
2830     local pth, ith
2831     for i=1,#path do
2832         pth = path[i]
2833         if not ith then
2834             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
2835         elseif curved(ith,pth) then
2836             local a, b = concat(ith.right_x,ith.right_y)

```

```

2837     local c, d = concat(pth.left_x, pth.left_y)
2838     pdf_literalcode("%f %f %f %f %f %f c", a, b, c, d, concat(pth.x_coord, pth.y_coord))
2839   else
2840     pdf_literalcode("%f %f l", concat(pth.x_coord, pth.y_coord))
2841   end
2842   ith = pth
2843 end
2844 if not open then
2845   local one = path[1]
2846   if curved(pth, one) then
2847     local a, b = concat(pth.right_x, pth.right_y)
2848     local c, d = concat(one.left_x, one.left_y)
2849     pdf_literalcode("%f %f %f %f %f %f c", a, b, c, d, concat(one.x_coord, one.y_coord))
2850   else
2851     pdf_literalcode("%f %f l", concat(one.x_coord, one.y_coord))
2852   end
2853 elseif #path == 1 then -- special case .. draw point
2854   local one = path[1]
2855   pdf_literalcode("%f %f l", concat(one.x_coord, one.y_coord))
2856 end
2857 end
2858

```

Finally, flush figures by inserting PDF literals.

```

2859 local function flush (result, flusher)
2860   if result then
2861     local figures = result.fig
2862     if figures then
2863       for f=1, #figures do
2864         info("flushing figure %s", f)
2865         local figure = figures[f]
2866         local objects = figure:objects()
2867         local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
2868         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2869         local bbox = figure:boundingbox()
2870         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
2871         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()

2872     else

```

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

```

2873     if tex_code_pre_mplib[f] then
2874       put2output(tex_code_pre_mplib[f])
2875     end

```

```

2876 pdf_startfigure(fignum,llx,lly,urx,ury)
2877 start_pdf_code()
2878 if objects then
2879   local savedpath = nil
2880   local savedhtap = nil
2881   for o=1,#objects do
2882     local object      = objects[o]
2883     local objecttype  = object.type

```

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

```

2884   local prescript      = object.prescript
2885   prescript = prescript and script2table(prescript) -- prescript is now a table
2886   local cr_over = do_preobj_CR(object,prescript) -- color
2887   local tr_opaq = do_preobj_TR(object,prescript) -- opacity
2888   local fading_ = do_preobj_FADE(object,prescript) -- fading
2889   local trgroup = do_preobj_GRP(object,prescript) -- transparency group
2890   local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
2891   local shading_ = do_preobj_shading(object,prescript,savedpath) -- shading pattern
2892   if prescript and prescript.mplibtexboxid then
2893     put_tex_boxes(object,prescript)
2894   elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
2895   elseif objecttype == "start_clip" then
2896     local evenodd = not object.istext and object.postscript == "evenodd"
2897     start_pdf_code()
2898     flushnormalpath(object.path,false)
2899     pdf_literalcode(evenodd and "W* n" or "W n")
2900   elseif objecttype == "stop_clip" then
2901     stop_pdf_code()
2902     miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2903   elseif objecttype == "special" then

```

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

```

2904   if prescript and prescript.postmplibverbtx then
2905     figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
2906   end
2907   elseif objecttype == "text" then
2908     local ot = object.transform -- 3,4,5,6,1,2
2909     start_pdf_code()
2910     pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
2911     pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
2912     stop_pdf_code()
2913   elseif not trgroup and fading_ ~= "stop" then
2914     local evenodd, collect, both = false, false, false
2915     local postscript = object.postscript
2916     if not object.istext then
2917       if postscript == "evenodd" then
2918         evenodd = true
2919       elseif postscript == "collect" then
2920         collect = true
2921       elseif postscript == "both" then

```

```

2922         both = true
2923     elseif postscript == "eoboth" then
2924         evenodd = true
2925         both = true
2926     end
2927 end
2928 if collect then
2929     if not savedpath then
2930         savedpath = { object.path or false }
2931         savedhtap = { object.htap or false }
2932     else
2933         savedpath[#savedpath+1] = object.path or false
2934         savedhtap[#savedhtap+1] = object.htap or false
2935     end
2936 else

```

Removed from ConT<sub>E</sub>Xt general: color stuff.

```

2937         local ml = object.miterlimit
2938         if ml and ml ~= miterlimit then
2939             miterlimit = ml
2940             pdf_literalcode("%f M",ml)
2941         end
2942         local lj = object.linejoin
2943         if lj and lj ~= linejoin then
2944             linejoin = lj
2945             pdf_literalcode("%i j",lj)
2946         end
2947         local lc = object.linecap
2948         if lc and lc ~= linecap then
2949             linecap = lc
2950             pdf_literalcode("%i J",lc)
2951         end
2952         local dl = object.dash
2953         if dl then
2954             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
2955             if d ~= dashed then
2956                 dashed = d
2957                 pdf_literalcode(dashed)
2958             end
2959         elseif dashed then
2960             pdf_literalcode("[ ] 0 d")
2961             dashed = false
2962         end
2963         local path = object.path
2964         local transformed, penwidth = false, 1
2965         local open = path and path[1].left_type and path[#path].right_type
2966         local pen = object.pen
2967         if pen then
2968             if pen.type == 'elliptical' then

```



```

2969         transformed, penwidth = pen_characteristics(object) -- boolean, value
2970         pdf_literalcode("%f w",penwidth)
2971         if objecttype == 'fill' then
2972             objecttype = 'both'
2973         end
2974     else -- calculated by mplib itself
2975         objecttype = 'fill'
2976     end
2977 end

```

Added : shading

```

2978     local shade_no, shade_stroking = do_preobj_SH(object,prescript) -- shading
2979     if shade_no then
2980         pdf_literalcode"q /Pattern cs"
2981         objecttype = false
2982     end
2983     if transformed then
2984         start_pdf_code()
2985     end
2986     if path then
2987         if savedpath then
2988             for i=1,#savedpath do
2989                 local path = savedpath[i]
2990                 if transformed then
2991                     flushconcatpath(path,open)
2992                 else
2993                     flushnormalpath(path,open)
2994                 end
2995             end
2996             savedpath = nil
2997         end
2998         if transformed then
2999             flushconcatpath(path,open)
3000         else
3001             flushnormalpath(path,open)
3002         end
3003         if objecttype == "fill" then
3004             pdf_literalcode(evenodd and "h f*" or "h f")
3005         elseif objecttype == "outline" then
3006             if both then
3007                 pdf_literalcode(evenodd and "h B*" or "h B")
3008             else
3009                 pdf_literalcode(open and "S" or "h S")
3010             end
3011         elseif objecttype == "both" then
3012             pdf_literalcode(evenodd and "h B*" or "h B")
3013         end
3014     end
3015     if transformed then

```

```

3016         stop_pdf_code()
3017     end
3018     local path = object.htap

```

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

```

3019     if path then
3020         if transformed then
3021             start_pdf_code()
3022         end
3023         if savedhtap then
3024             for i=1,#savedhtap do
3025                 local path = savedhtap[i]
3026                 if transformed then
3027                     flushconcatpath(path,open)
3028                 else
3029                     flushnormalpath(path,open)
3030                 end
3031             end
3032             savedhtap = nil
3033             evenodd = true
3034         end
3035         if transformed then
3036             flushconcatpath(path,open)
3037         else
3038             flushnormalpath(path,open)
3039         end
3040         if objecttype == "fill" then
3041             pdf_literalcode(evenodd and "h f*" or "h f")
3042         elseif objecttype == "outline" then
3043             pdf_literalcode(open and "S" or "h S")
3044         elseif objecttype == "both" then
3045             pdf_literalcode(evenodd and "h B*" or "h B")
3046         end
3047         if transformed then
3048             stop_pdf_code()
3049         end
3050     end

```

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

```

3051         if shade_no then -- shading
3052             pdf_literalcode("W%s %s /MPLibSh%s sh Q",
3053                 evenodd and "*" or "", shade_stroking == "yes" and "s" or "n", shade_no)
3054         end
3055     end
3056 end
3057 if fading_ == "start" then
3058     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}
3059 elseif trgroup == "start" then
3060     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
3061 elseif fading_ == "stop" then

```

```

3062         local se = pdfetcs.fading.specialeffects
3063         if se then stop_special_effects(se[1], se[2], se[3]) end
3064     elseif trgroup == "stop" then
3065         local se = pdfetcs.tr_group.specialeffects
3066         if se then stop_special_effects(se[1], se[2], se[3]) end
3067     else
3068         stop_special_effects(fading_, tr_opaq, cr_over)
3069     end
3070     if fading_ or trgroup then -- extgs resetted
3071         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3072     end
3073 end
3074 end
3075 stop_pdf_code()
3076 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtex code.

```

3077     for _,v in ipairs(figcontents) do
3078         if type(v) == "table" then
3079             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3080         else
3081             texsprint(v)
3082         end
3083     end
3084     if #figcontents.post > 0 then texsprint(figcontents.post) end
3085     figcontents = { post = { } }
3086 end
3087 end
3088 end
3089 end
3090 end
3091
3092 function luamplib.convert (result, flusher)
3093     flush(result, flusher)
3094     return true -- done
3095 end
3096 end
3097
3098 function luamplib.colorconverter (cr)
3099     local n = #cr
3100     if n == 4 then
3101         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3102         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3103     elseif n == 3 then
3104         local r, g, b = cr[1], cr[2], cr[3]
3105         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3106     else
3107         local s = cr[1]
3108         return format("%.3f g %.3f G",s,s), "0 g 0 G"

```

```

3109 end
3110 end

```

## 2.2 $\TeX$ package

First we need to load some packages.

```

3111 \ifcsname ProvidesPackage\endcsname

```

We need  $\LaTeX$  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.

```

3112 \NeedsTeXFormat{LaTeX2e}
3113 \ProvidesPackage{luamplib}
3114 [2026/01/27 v2.38.3 mplib package for LuaTeX]
3115 \fi
3116 \ifdefined\newluafunction\else
3117 \input ltluatex
3118 \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  $\LaTeX$  kernel. In Plain, `atbegshi.sty` is loaded.

```

3119 \ifnum\outputmode=0
3120 \ifdefined\AddToHookNext
3121 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3122 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3123 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3124 \else
3125 \input atbegshi.sty
3126 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3127 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3128 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3129 \fi
3130 \fi

```

Loading of lua code.

```

3131 \directlua{require("luamplib")}

```

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

```

3132 \ifx\pdfoutput\undefined
3133 \let\pdfoutput\outputmode
3134 \fi
3135 \ifx\pdfliteral\undefined
3136 \protected\def\pdfliteral{\pdfextension literal}
3137 \fi

```

Set the format for `METAPOST`.

```

3138 \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.

```

3139 \ifnum\pdfoutput>0
3140   \let\mplibtoPDF\pdfliteral
3141 \else
3142   \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3143   \ifcsname PackageInfo\endcsname
3144     \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3145   \else
3146     \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3147   \fi
3148 \fi

```

To make `mplibcode` typeset always in horizontal mode.

```

3149 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3150 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3151 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in `mplibcode`.

```

3152 \def\mplibsetupcatcodes{%
3153   %catcode`\{=12 %catcode`\}=12
3154   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3155   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^M=12
3156 }

```

Make `btex...etex` box zero-metric.

```

3157 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}

```

use Transparency Group

```

3158 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3159 \def\usemplibgroupmain#1{%
3160   \prependtomplibbox\hbox dir TLT\bgroup
3161   \csname luamplib.group.#1\endcsname
3162   \egroup
3163 }
3164 \protected\def\mplibgroup#1{%
3165   \begingroup
3166   \def\MPllx{0}\def\MPlly{0}%
3167   \def\mplibgroupname{#1}%
3168   \mplibgroupgetnexttok
3169 }
3170 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3171 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
3172 \def\mplibgroupbranch{%
3173   \ifx [\nexttok
3174     \expandafter\mplibgroupopts
3175   \else
3176     \ifx\mplibsptoken\nexttok
3177       \expandafter\expandafter\expandafter\mplibgroupskipspace
3178     \else
3179       \let\mplibgroupoptions\empty
3180       \expandafter\expandafter\expandafter\mplibgroupmain
3181     \fi

```

```

3182 \fi
3183 }
3184 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3185 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3186 \protected\def\endmplibgroup{\egroup
3187 \directlua{ luamplib.registergroup(
3188 \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3189 )}%
3190 \endgroup
3191 }

Patterns

3192 {\def\:\global\let\mplibsptoken= } \: }
3193 \protected\def\mppattern#1{%
3194 \begingroup
3195 \def\mplibpatternname{#1}%
3196 \mplibpatterngetnexttok
3197 }
3198 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3199 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3200 \def\mplibpatternbranch{%
3201 \ifx [\nexttok
3202 \expandafter\mplibpatternopts
3203 \else
3204 \ifx\mplibsptoken\nexttok
3205 \expandafter\expandafter\expandafter\mplibpatternskipsspace
3206 \else
3207 \let\mplibpatternoptions\empty
3208 \expandafter\expandafter\expandafter\mplibpatternmain
3209 \fi
3210 \fi
3211 }
3212 \def\mplibpatternopts[#1]{%
3213 \def\mplibpatternoptions{#1}%
3214 \mplibpatternmain
3215 }
3216 \def\mplibpatternmain{%
3217 \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3218 }
3219 \protected\def\endmppattern{%
3220 \egroup
3221 \directlua{ luamplib.registerpattern(
3222 \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3223 )}%
3224 \endgroup
3225 }

simple way to use mplib: \mpfig draw fullcircle scaled 10; \endmpfig
3226 \def\mpfiginstancename{@mpfig}
3227 \protected\def\mpfig{%

```

```

3228 \begingroup
3229 \futurelet\nexttok\mplibmpfigbranch
3230 }
3231 \def\mplibmpfigbranch{%
3232   \ifx *\nexttok
3233     \expandafter\mplibprempfig
3234   \else
3235     \ifx [\nexttok
3236       \expandafter\expandafter\expandafter\mplibgobbleoptsmpfig
3237     \else
3238       \expandafter\expandafter\expandafter\mplibmainmpfig
3239     \fi
3240   \fi
3241 }
3242 \def\mplibgobbleoptsmpfig[#1]{\mplibmainmpfig}
3243 \def\mplibmainmpfig{%
3244   \begingroup
3245   \mplibsetupcatcodes
3246   \mplibdomainmpfig
3247 }
3248 \long\def\mplibdomainmpfig#1\endmpfig{%
3249   \endgroup
3250   \directlua{
3251     local legacy = luamplib.legacyverbatim
3252     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3253     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3254     luamplib.legacyverbatim = false
3255     luamplib.everymplib["\mpfiginstancename"] = ""
3256     luamplib.everyendmplib["\mpfiginstancename"] = ""
3257     luamplib.process_mplibcode(
3258       "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===].." "..everyendmpfig.." endfig;",
3259       "\mpfiginstancename")
3260     luamplib.legacyverbatim = legacy
3261     luamplib.everymplib["\mpfiginstancename"] = everympfig
3262     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3263   }%
3264   \endgroup
3265 }
3266 \def\mplibprempfig#1{%
3267   \begingroup
3268   \mplibsetupcatcodes
3269   \mplibdoprempfig
3270 }
3271 \long\def\mplibdoprempfig#1\endmpfig{%
3272   \endgroup
3273   \directlua{
3274     local legacy = luamplib.legacyverbatim
3275     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3276     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]

```

```

3277   luamplib.legacyverbatimimtex = false
3278   luamplib.everymplib["\mpfiginstancename"] = ""
3279   luamplib.everyendmplib["\mpfiginstancename"] = ""
3280   luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
3281   luamplib.legacyverbatimimtex = legacy
3282   luamplib.everymplib["\mpfiginstancename"] = everympfig
3283   luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3284 }%
3285 \endgroup
3286 }
3287 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3288 \unless\ifcsname ver@luamplib.sty\endcsname
3289   \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3290   \protected\def\mplibcode{%
3291     \begingroup
3292     \futurelet\nexttok\mplibcodebranch
3293   }
3294   \def\mplibcodebranch{%
3295     \ifx [\nexttok
3296       \expandafter\mplibcodegetinstancename
3297     \else
3298       \global\let\currentmpinstancename\empty
3299       \expandafter\mplibcodeindeed
3300     \fi
3301   }
3302   \def\mplibcodeindeed{%
3303     \begingroup
3304     \mplibsetupcatcodes
3305     \mplibdocode
3306   }
3307   \long\def\mplibdocode#1\endmplibcode{%
3308     \endgroup
3309     \directlua{luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\currentmpinstancename")}%
3310     \endgroup
3311   }
3312   \protected\def\endmplibcode{endmplibcode}
3313 \else

```

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

```

3314   \newenvironment{mplibcode}[1][{}]{%
3315     \xdef\currentmpinstancename{#1}%
3316     \mplibtmp toks{}\ltxdomplibcode
3317   }{}
3318   \def\ltxdomplibcode{%
3319     \begingroup
3320     \mplibsetupcatcodes
3321     \ltxdomplibcodeindeed
3322   }

```



```

3323 \def\mplib@mplibcode{mplibcode}
3324 \long\def\ltxdomplibcodeindeed#1\end#2{%
3325   \endgroup
3326   \mplibtmp toks\expandafter{\the\mplibtmp toks#1}%
3327   \def\mplibtemp@a{#2}%
3328   \ifx\mplib@mplibcode\mplibtemp@a
3329     \directlua{luamplib.process_mplibcode([==[\the\mplibtmp toks]==], "\currentmpinstancename")}%
3330     \end{mplibcode}%
3331   \else
3332     \mplibtmp toks\expandafter{\the\mplibtmp toks\end{#2}}%
3333     \expandafter\ltxdomplibcode
3334   \fi
3335 }
3336 \fi

```

User settings.

```

3337 \def\mplibshowlog#1{\directlua{
3338   local s = string.lower("#1")
3339   if s == "enable" or s == "true" or s == "yes" then
3340     luamplib.showlog = true
3341   else
3342     luamplib.showlog = false
3343   end
3344 }}
3345 \def\mpliblegacybehavior#1{\directlua{
3346   local s = string.lower("#1")
3347   if s == "enable" or s == "true" or s == "yes" then
3348     luamplib.legacyverbatim = true
3349   else
3350     luamplib.legacyverbatim = false
3351   end
3352 }}
3353 \def\mplibverbatim#1{\directlua{
3354   local s = string.lower("#1")
3355   if s == "enable" or s == "true" or s == "yes" then
3356     luamplib.verbatiminput = true
3357   else
3358     luamplib.verbatiminput = false
3359   end
3360 }}
3361 \newtoks\mplibtmp toks

```

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

```

3362 \ifcsname ver@luamplib.sty\endcsname
3363   \protected\def\everymplib{%
3364     \begingroup
3365     \mplibsetupcatcodes
3366     \mplibdoeverymplib
3367   }
3368   \protected\def\everyendmplib{%

```

```

3369 \begingroup
3370 \mplibsetupcatcodes
3371 \mplibdoeveryendmplib
3372 }
3373 \newcommand\mplibdoeverymplib[2][{}]{%
3374 \endgroup
3375 \directlua{
3376   luampplib.everymplib["#1"] = [===[\unexpanded{#2}]===[
3377   ]%
3378 }
3379 \newcommand\mplibdoeveryendmplib[2][{}]{%
3380 \endgroup
3381 \directlua{
3382   luampplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===[
3383   ]%
3384 }
3385 \else
3386 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3387 \protected\def\everymplib#1#%
3388 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3389 \begingroup
3390 \mplibsetupcatcodes
3391 \mplibdoeverymplib
3392 }
3393 \long\def\mplibdoeverymplib#1{%
3394 \endgroup
3395 \directlua{
3396   luampplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===[
3397   ]%
3398 }
3399 \protected\def\everyendmplib#1#%
3400 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3401 \begingroup
3402 \mplibsetupcatcodes
3403 \mplibdoeveryendmplib
3404 }
3405 \long\def\mplibdoeveryendmplib#1{%
3406 \endgroup
3407 \directlua{
3408   luampplib.everyendmplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===[
3409   ]%
3410 }
3411 \fi

```

**T<sub>E</sub>X** macros for dimen/color

```

3412 \def\mpdim#1{ runscript("luampplibdimen{#1}") }
3413 \def\mpcolor#1#{\domplibcolor{#1}}
3414 \def\domplibcolor#1#2{ runscript("luampplibcolor{#1{#2}}") }

```

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

```

3415 \def\mplibnumbersystem#1{\directlua{
3416   local t = "#1"
3417   if t == "binary" then t = "decimal" end
3418   luamplib.numbersystem = t
3419 }}

```

Settings for .mp cache files.

```

3420 \def\mplibmakenocache#1{\mplibdomakenocache #1,*}
3421 \def\mplibdomakenocache#1,{%
3422   \ifx\empty#1\empty
3423     \expandafter\mplibdomakenocache
3424   \else
3425     \ifx*#1\else
3426       \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
3427       \expandafter\expandafter\expandafter\mplibdomakenocache
3428     \fi
3429   \fi
3430 }
3431 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*}
3432 \def\mplibdocancelnocache#1,{%
3433   \ifx\empty#1\empty
3434     \expandafter\mplibdocancelnocache
3435   \else
3436     \ifx*#1\else
3437       \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
3438       \expandafter\expandafter\expandafter\mplibdocancelnocache
3439     \fi
3440   \fi
3441 }
3442 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

3443 \def\mplibtexttextlabel#1{\directlua{
3444   local s = string.lower("#1")
3445   if s == "enable" or s == "true" or s == "yes" then
3446     luamplib.texttextlabel = true
3447   else
3448     luamplib.texttextlabel = false
3449   end
3450 }}
3451 \def\mplibcodeinherit#1{\directlua{
3452   local s = string.lower("#1")
3453   if s == "enable" or s == "true" or s == "yes" then
3454     luamplib.codeinherit = true
3455   else
3456     luamplib.codeinherit = false
3457   end
3458 }}
3459 \def\mplibglobaltexttext#1{\directlua{
3460   local s = string.lower("#1")

```

```

3461   if s == "enable" or s == "true" or s == "yes" then
3462     luamplib.globaltexttext = true
3463   else
3464     luamplib.globaltexttext = false
3465   end
3466 }}

```

The followings are from ConT<sub>E</sub>Xt general, mostly.

We use a dedicated scratchbox.

```

3467 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

3468 \def\mplibstarttoPDF#1#2#3#4{%
3469   \prependtomplibbox
3470   \hbox dir TLT\bgroup
3471   \xdef\MPllx{#1}\xdef\MPlly{#2}%
3472   \xdef\MPurx{#3}\xdef\MPury{#4}%
3473   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3474   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3475   \parskip0pt%
3476   \leftskip0pt%
3477   \parindent0pt%
3478   \everypar{}%
3479   \setbox\mplibscratchbox\vbox\bgroup
3480   \noindent
3481 }
3482 \def\mplibstoptoPDF{%
3483   \par
3484   \egroup %
3485   \setbox\mplibscratchbox\hbox %
3486     {\hskip-\MPllx bp%
3487      \raise-\MPlly bp%
3488      \box\mplibscratchbox}%
3489   \setbox\mplibscratchbox\vbox to \MPheight
3490     {\vfill
3491      \hsize\MPwidth
3492      \wd\mplibscratchbox0pt%
3493      \ht\mplibscratchbox0pt%
3494      \dp\mplibscratchbox0pt%
3495      \box\mplibscratchbox}%
3496   \wd\mplibscratchbox\MPwidth
3497   \ht\mplibscratchbox\MPheight
3498   \box\mplibscratchbox
3499   \egroup
3500 }

```

Text items have a special handler.

```

3501 \def\mplibtexttext#1#2#3#4#5{%
3502   \begingroup
3503   \setbox\mplibscratchbox\hbox

```

```

3504    {\font\temp=#1 at #2bp%
3505      \temp
3506      #3}%
3507    \setbox\mplibscratchbox\hbox
3508      {\hskip#4 bp%
3509        \raise#5 bp%
3510        \box\mplibscratchbox}%
3511    \wd\mplibscratchbox0pt%
3512    \ht\mplibscratchbox0pt%
3513    \dp\mplibscratchbox0pt%
3514    \box\mplibscratchbox
3515    \endgroup
3516 }

```

Input luamplib.cfg when it exists.

```

3517 \openin0=luamplib.cfg
3518 \ifeof0 \else
3519   \closein0
3520   \input luamplib.cfg
3521 \fi

```

Code for tagpdf

```

3522 \def\luamplibtagtextboxset#1#2{#2}
3523 \let\luamplibnotagtextboxset\luamplibtagtextboxset
3524 \let\luamplibtagasgroupset\relax
3525 \let\luamplibtagasgroupput\luamplibtagtextboxset
3526 \ifcsname SuspendTagging\endcsname\else\endinput\fi
3527 \ifcsname ver@tagpdf.sty\endcsname \else
3528   \ExplSyntaxOn
3529   \keys_define:nn{luamplib/tagging}
3530   {
3531     ,alt          .code:n = { }
3532     ,actualtext   .code:n = { }
3533     ,artifact     .code:n = { }
3534     ,text         .code:n = { }
3535     ,off          .code:n = { }
3536     ,tag          .code:n = { }
3537     ,adjust-BBox  .code:n = { }
3538     ,tagging-setup .code:n = { }
3539     ,instance     .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3540     ,instancename .meta:n = { instance = {#1} }
3541     ,unknown      .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
3542   }
3543   \RenewDocumentCommand\mplibcode{0{}}
3544   {
3545     \tl_gclear:N \currentmpinstancename
3546     \keys_set:ne{luamplib/tagging}{#1}
3547     \mplibtmptoks{}\ltxdomplibcode
3548   }
3549   \cs_set_eq:NN \mplibaltext \use_none:n

```

3550 \cs\_set\_eq:NN \mplibactualtext \use\_none:n

2025/12/05: \begin{center}\mpfig ... \endmpfig\end{center} raises an Error! as we issue \everypar{} before flushing literals out. It is related to \partokencontext=2 recently introduced by L<sup>A</sup>T<sub>E</sub>X. Why we used vbox initially? where hbox seems to be sufficient. Anyway, among various solutions including \partokencontext\z@, \let\par\@@par, and \endgraf, we here attempt to address the issue by adding the following line, which L<sup>A</sup>T<sub>E</sub>X's \everypar should have done.

```

3551 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
3552 \ExplSyntaxOff
3553 \endinput\fi
3554 \ExplSyntaxOn
3555 \tl_new:N \l__luamplib_tag_envname_tl
3556 \tl_new:N \l__luamplib_tag_alt_tl
3557 \tl_new:N \l__luamplib_tag_alt_dflt_tl
3558 \tl_new:N \l__luamplib_tag_actual_tl
3559 \tl_new:N \l__luamplib_tag_struct_tl
3560 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
3561 \bool_new:N \l__luamplib_tag_usetext_bool
3562 \bool_new:N \l__luamplib_tag_bboxcorr_bool
3563 \seq_new:N \l__luamplib_tag_bboxcorr_seq
3564 \tl_new:N \l__luamplib_tag_bbox_draw_tl
3565 \tl_new:N \l__luamplib_BBox_llx_tl
3566 \tl_new:N \l__luamplib_BBox_lly_tl
3567 \tl_new:N \l__luamplib_BBox_urx_tl
3568 \tl_new:N \l__luamplib_BBox_ury_tl
3569 \msg_new:nnn {luamplib}{figure-text-reuse}
3570 {
3571   tex-text~box~#1~probably~is~incorrectly~tagged.~
3572   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
3573   Check~the~resulting~PDF.
3574 }
3575 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
3576 {
3577   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
3578   Using~mplibgroup~with~text~mode~is~not~recommended.~
3579   Check~the~resulting~PDF.
3580 }
3581 \msg_new:nnn{luamplib}{alt-text-missing}
3582 {
3583   Alternate~text~for~#1~is~missing.~
3584   Using~the~default~value~'#2'~instead.
3585 }

```

Sockets for tex-text boxes.

```

3586 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
3587 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
3588 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
3589 {

```

TODO: we check text mode here. If we tag text boxes for all modes, we will get a lot of

structure-has-no-parent warning; no good-looking, though it seems to be no harm.

```

3590 \bool_if:NTF \l__luamplib_tag_usetext_bool
3591 {
3592   \tag_mc_end_push:
3593   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
3594   \cs_gset_nopar:cpe {luamplib.taggedbox.#1} {\tag_get:n{struct_num}}

```

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

```

3595   \tag_mc_begin:n{tag=text}
3596   #2
3597   \tag_mc_end:
3598   \tag_struct_end:
3599   \tag_mc_begin_pop:n{ }
3600 }
3601 {
3602   \tag_suspend:n{\luamplibtagtextboxset}
3603   #2
3604   \tag_resume:n{\luamplibtagtextboxset}
3605 }
3606 }
3607 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
3608 {
3609   \bool_lazy_and:nnTF
3610   { \l__luamplib_tag_usetext_bool }
3611   { \cs_if_free_p:c {luamplib.notaggedbox.#1} }
3612   {
3613     \tag_resume:n{\mplibputtextbox}
3614     \tag_mc_end:
3615     \cs_if_exist:cTF {luamplib.taggedbox.#1}
3616     {
3617       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
3618       #2
3619       \cs_undefine:c {luamplib.taggedbox.#1}
3620     }
3621     {
3622       \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
3623       \tag_mc_begin:n{ }
3624       \int_set:Nn \l_tmpa_int {#1}
3625       \tag_mc_reset_box:N \l_tmpa_int
3626       #2
3627       \tag_mc_end:
3628     }
3629     \tag_mc_begin:n{artifact}
3630   }
3631   {
3632     \int_set:Nn \l_tmpa_int {#1}
3633     \tag_mc_reset_box:N \l_tmpa_int
3634     #2
3635   }

```

```

3636 }
3637 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
3638 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
3639 \cs_set_nopar:Npn \luamplibtagtextboxset
3640 {
3641   \tag_socket_use:nnn{luamplib/texttext/set}
3642 }

```

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

```

3643 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
3644 {
3645   \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
3646   \bool_set_false:N \l__luamplib_tag_usetext_bool
3647   \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
3648   \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
3649   \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
3650 }
3651 \cs_set_nopar:Npn \mplibputtextbox #1
3652 {
3653   \vbox to 0pt{\vss\hbox to 0pt{
3654     \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}
3655     \hss}}
3656 }

```

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

```

3657 \cs_set_nopar:Npn \luamplibtagasgroupset
3658 {
3659   \bool_set_false:N \l__luamplib_tag_usetext_bool
3660 }
3661 \cs_set_nopar:Npn \luamplibtagasgroupput
3662 {
3663   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
3664   \tag_socket_use:nnn{luamplib/mplibgroup/put}
3665 }

```

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

```

3666 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
3667 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
3668 {
3669   \cs_if_free:cT {luamplib.mplibgroup.text.#1}
3670   {
3671     \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
3672     \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
3673   }
3674   \tag_mc_end:
3675   \tag_mc_begin:n{tag=text}
3676   #2
3677   \tag_mc_end:

```



```

3678 \tag_mc_begin:n{artifact}
3679 }
3680 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

#### A macro for BBox attribute

```

3681 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
3682 {
3683   \tl_set:Nx \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
3684   \tex_savepos:D
3685   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
3686   \tl_set:Nx \l__luamplib_BBox_llx_tl
3687     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
3688   \tl_set:Nx \l__luamplib_BBox_lly_tl
3689     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
3690   \tl_set:Nx \l__luamplib_BBox_urx_tl
3691     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
3692   \tl_set:Nx \l__luamplib_BBox_ury_tl
3693     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
3694   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
3695   {
3696     \int_zero:N \l_tmpa_int
3697     \tl_map_inline:nn
3698     {
3699       \l__luamplib_BBox_llx_tl
3700       \l__luamplib_BBox_lly_tl
3701       \l__luamplib_BBox_urx_tl
3702       \l__luamplib_BBox_ury_tl
3703     }
3704     {
3705       \int_incr:N \l_tmpa_int
3706       \tl_set:Nx ##1
3707       {
3708         \fp_eval:n
3709         {
3710           ##1
3711           +
3712           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
3713         }
3714       }
3715     }
3716   }
3717   \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
3718   {
3719     /O /Layout /BBox [
3720       \l__luamplib_BBox_llx_tl\c_space_tl
3721       \l__luamplib_BBox_lly_tl\c_space_tl
3722       \l__luamplib_BBox_urx_tl\c_space_tl
3723       \l__luamplib_BBox_ury_tl
3724     ]

```

```

3725 }
3726 \bool_if:NT \l__tag_graphic_debug_bool
3727 {
3728   \iow_log:e
3729   {
3730     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
3731     \l__luamplib_BBox_llx_tl\c_space_tl
3732     \l__luamplib_BBox_lly_tl\c_space_tl
3733     \l__luamplib_BBox_urx_tl\c_space_tl
3734     \l__luamplib_BBox_ury_tl
3735   }
3736   \sys_if_output_pdf:TF
3737   {
3738     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
3739     {
3740       \pdfextension save\relax
3741       \opacity_select:n{0.5} \color_select:n{red}
3742       \pdfextension literal~text
3743       {
3744         \l__luamplib_BBox_llx_tl\c_space_tl
3745         \l__luamplib_BBox_lly_tl\c_space_tl
3746         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3747         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3748         re~f
3749       }
3750       \pdfextension restore\relax
3751     }
3752   }
3753   {
3754     \tl_set:Nx \l__luamplib_tag_bbox_draw_tl
3755     {
3756       \special{pdf:bcontent}
3757       \opacity_select:n{0.5} \color_select:n{red}
3758       \special{pdf:code~
3759         1~0~0~1~
3760         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
3761         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
3762         cm
3763       }
3764       \special{pdf:code~
3765         \l__luamplib_BBox_llx_tl\c_space_tl
3766         \l__luamplib_BBox_lly_tl\c_space_tl
3767         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3768         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3769         re~f
3770       }
3771       \special{pdf:econtent}
3772     }
3773   }

```

```

3774 }
3775 }

```

## Sockets for main process

```

3776 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
3777 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
3778 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
3779 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
3780 {
3781   \tag_mc_end_push:
3782   \tl_if_empty:NT\l__luamplib_tag_alt_tl
3783   {
3784     \tl_if_empty:eTF{#1}
3785     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
3786     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
3787     \msg_warning:nnVV{luamplib}{alt-text-missing}
3788     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
3789   }
3790   \tag_struct_begin:n
3791   {
3792     tag=\l__luamplib_tag_struct_tl,
3793     alt=\l__luamplib_tag_alt_tl,
3794   }
3795   \tag_mc_begin:n{ }
3796 }
3797 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
3798 {
3799   \__luamplib_tag_bbox_attribute:n {#1}
3800   #2
3801   \tl_use:N \l__luamplib_tag_bbox_draw_tl
3802   \tag_mc_end:
3803   \tag_struct_end:
3804   \tag_mc_begin_pop:n{ }
3805 }
3806 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
3807 {
3808   \tag_mc_end_push:
3809   \tag_struct_begin:n
3810   {
3811     tag=Span,
3812     actualtext=\l__luamplib_tag_actual_tl,
3813   }
3814   \tag_mc_begin:n{ }
3815 }
3816 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
3817 {
3818   #2
3819   \tag_mc_end:
3820   \tag_struct_end:

```

```

3821 \tag_mc_begin_pop:n{ }
3822 }
3823 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
3824 {
3825 \tag_mc_end_push:
3826 \tag_mc_begin:n{artifact}
3827 }
3828 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
3829 {
3830 #2
3831 \tag_mc_end:
3832 \tag_mc_begin_pop:n{ }
3833 }

```

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

```

3834 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
3835 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
3836 {
3837 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
3838 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
3839 }
3840 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
3841 {
3842 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
3843 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

3844 \prependtomplibbox \mplibnoforcehmode
3845 \mode_if_vertical:T { \noindent \aftergroup\par }
3846 }
3847 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
3848 {
3849 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
3850 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
3851 }
3852 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
3853 {
3854 \bool_set_true:N \l__luamplib_tag_usetext_bool
3855 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
3856 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
3857 \prependtomplibbox \mplibnoforcehmode
3858 \mode_if_vertical:T { \noindent \aftergroup\par }
3859 }
3860 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
3861 {
3862 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
3863 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
3864 }
3865 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

## Key-value options

```

3866 \keys_define:nn{luamplib/tagging}
3867 {
3868   ,alt .code:n =
3869   {
3870     \tl_set:Nn\__luamplib_tag_alt_tl{\text_purify:n{#1}}
3871     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
3872   }
3873   ,actualtext .code:n =
3874   {
3875     \tl_set:Nn\__luamplib_tag_actual_tl{\text_purify:n{#1}}
3876     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
3877   }
3878   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
3879   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
3880   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
3881   ,tag .code:n =
3882   {
3883     \str_case:nnF {#1}
3884     {
3885       {false} { \keys_set:nn {luamplib/tagging} {off} }
3886       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
3887     }
3888     {
3889       \tl_set:Nn\__luamplib_tag_struct_tl{#1}
3890       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
3891     }
3892   }
3893   ,adjust-BBox .code:n =
3894   {
3895     \bool_set_true:N \__luamplib_tag_bboxcorr_bool
3896     \seq_set_split:Nnn \__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
3897   }
3898   ,tagging-setup .code:n = { \keys_set:nn {luamplib/tagging} {#1} }
3899 }
3900 \keys_define:nn {luamplib/instance}
3901 {
3902   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3903   ,instancename .meta:n = { instance = {#1} }
3904   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
3905 }

```

## Redefine our macros

```

3906 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
3907 {
3908   \prependtomplibbox
3909   \hbox dir~TLT\bgroup
3910     \tag_socket_use:nn{luamplib/figure/begin}\__luamplib_tag_alt_dflt_tl
3911     \xdef\MPl1x{#1}\xdef\MPlly{#2}%

```

```

3912 \xdef\MPurx{#3}\xdef\MPury{#4}%
3913 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3914 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3915 \parskip0pt
3916 \leftskip0pt
3917 \parindent0pt
3918 \everypar{}%
3919 \setbox\mplibscratchbox\vbox\bgroup
3920 \tag_suspend:n{\mplibstarttoPDF}
3921 \noindent
3922 }
3923 \cs_set_nopar:Npn \mplibstoptoPDF
3924 {
3925 \par
3926 \egroup
3927 \setbox\mplibscratchbox\hbox
3928 {\hskip-\MPllx bp
3929 \raise-\MPlly bp
3930 \box\mplibscratchbox}%
3931 \setbox\mplibscratchbox\vbox to \MPheight
3932 {\vfill
3933 \hsize\MPwidth
3934 \wd\mplibscratchbox0pt
3935 \ht\mplibscratchbox0pt
3936 \dp\mplibscratchbox0pt
3937 \box\mplibscratchbox}%
3938 \wd\mplibscratchbox\MPwidth
3939 \ht\mplibscratchbox\MPheight
3940 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
3941 \egroup
3942 }
3943 \RenewDocumentCommand\mplibcode{0{}}
3944 {
3945 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
3946 \tl_gclear:N \currentmpinstancename
3947 \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
3948 \keys_set:nV {luamplib/instance} \l_tmpa_tl
3949 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
3950 \tag_socket_use:n{luamplib/figure/init}
3951 \mplibtmptoks{}\ltxdomplibcode
3952 }
3953 \RenewDocumentCommand\mpfig{s 0{}}
3954 {
3955 \begingroup
3956 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
3957 \keys_set_known:ne {luamplib/tagging} {#2}
3958 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
3959 \tag_socket_use:n{luamplib/figure/init}
3960 \IfBooleanTF{#1} { \mplibprempfig * }

```

```

3961             { \mplibmainmpfig }
3962 }
3963 \RenewDocumentCommand\usemplibgroup{0}{ m}
3964 {
3965   \begingroup
3966   \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
3967   \keys_set_known:ne {luamplib/tagging} {#1}
3968   \tag_socket_use:n{luamplib/figure/init}
3969   \prependtomplibbox\hbox dir~TLT\bgroup
3970     \tag_socket_use:nn{luamplib/figure/begin}{#2}
3971     \setbox\mplibscratchbox\hbox\bgroup
3972       \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
3973       \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
3974       \egroup
3975       \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
3976     \egroup
3977   \endgroup
3978 }

```

Allow setting alt/actual text within METAPOST code. Of course we can use them in T<sub>E</sub>X code as well.

```

3979 \cs_new_nopar:Npn \mplibalttext #1
3980 {
3981   \tl_set:Ne \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
3982 }
3983 \cs_new_nopar:Npn \mplibactualtext #1
3984 {
3985   \tl_set:Ne \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
3986 }
3987 \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>Preamble</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. 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 licensed 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>TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION</div> <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><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><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</div></div>	<div>you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.</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 (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:</div> <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> <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>No WARRANTY</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 REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.</div> <div>END OF TERMS AND CONDITIONS</div> <div>Appendix: How to Apply These Terms to Your New Programs</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 show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program.</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>Yoyodyne, 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 subroutine 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>
---	---	---