

# The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/06/19 v2.42.2

## 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>	5
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	9
1.1.16	About figure box metric	9
1.1.17	<code>luamplib.cfg</code>	9
1.1.18	Tagged PDF	9
1.2	METAPOST	11
1.2.1	<code>mplibdimen, mplibcolor</code>	11
1.2.2	<code>mplibtexcolor, mplibrgbtexcolor</code>	11
1.2.3	<code>withmplibcolors</code>	12
1.2.4	<code>withtransparency</code>	12

1.2.5	<code>withmplibopacities</code>	13
1.2.6	<code>mplibgraphicstext</code>	13
1.2.7	<code>mplibglyph</code>	14
1.2.8	<code>mplibdrawglyph</code> , and its friends	14
1.2.9	<code>mpliboutlinetext</code>	15
1.2.10	<code>mpliblength</code> , <code>mplibuclength</code>	15
1.2.11	<code>mplibsubstring</code> , <code>mplibucsubstring</code>	15
1.2.12	<code>withshadingmethod</code>	16
1.2.13	<code>withfademethod</code>	18
1.2.14	<code>\mppattern</code> , <code>withmppattern</code>	19
1.2.15	<code>asgroup</code>	21
1.2.16	<code>\mplibgroup</code>	24
1.2.17	<code>withmaskinggroup</code>	25
1.2.18	Free-form Gouraud-shaded triangle mesh shading	26
1.2.19	Lattice-form Gouraud-shaded triangle mesh shading	28
1.2.20	Coons patch mesh shading	28
1.2.21	Tensor-product patch mesh shading	30
1.3	Lua	31
1.3.1	<code>runscript</code>	31
1.3.2	<code>luamplib.instances</code>	31
1.3.3	<code>luamplib.process_mplibcode</code>	32
1.3.4	<code>luamplib.registerpattern</code>	33
1.3.5	<code>luamplib.registergroup</code>	33
<b>2</b>	<b>Implementation</b>	<b>33</b>
2.1	Lua module	33
2.2	$\TeX$ package	114
<b>3</b>	<b>The GNU GPL License v2</b>	<b>135</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 <string>` is a more versatile macro equivalent to `TEX <string>` from `TEX.mp`. `TEX` is also allowed and is a synonym of `texttext`. The argument of `mplib`'s primitive `maketext` will also be processed by the same routine.
- Possibility to use `verbatimtex ... etex` to run a  $\TeX$  code. `VerbatimTeX <string>` is a more versatile macro corresponding to `verbatimtex` command. Of course the behavior cannot be the same as the stand-alone `mpost`, so that you cannot include `\documentclass`, `\usepackage` etc. When these  $\TeX$  commands are found in `verbatimtex ... etex`, the entire code will be ignored.

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

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

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

## 1.1 $\TeX$

### 1.1.1 `\mplibforcehmode`

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

---

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

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

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

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

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

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

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

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

where  $\langle \text{string} \rangle$  should be "" (empty), "isolated", "knockout", or "isolated,knockout". Beware that currently many of the PDF rendering applications, except Adobe Acrobat and  $\text{\TeX}$ works, cannot properly render the isolated or knockout effect.

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

#### 1.1.4 `\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 `META-POST` process will be printed to the `.log` file. This is the  $\text{\TeX}$  side interface for `luamplib.showlog`.

---

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

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


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

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

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

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

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

<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.

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

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

### 1.1.8 `\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  $\TeX$  environment `mplibcode` or Plain  $\TeX$  commands `\mplibcode` ... `\endmplibcode`. The syntax for  $\TeX$  is:

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

The behavior is as follows.

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

In 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` module is well supported in PDF and DVI mode. Package `colorspace` is supported as well in PDF mode, but could conflict with `luamplib`'s special features such as shading when `\DocumentMetadata`, i.e. PDF management code, is not loaded.

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

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

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



N.B. Be aware, however, that even after v2.38.1 `\mpcolor` still inserts `withprescript` command when the color specified is a spot color (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)
  scaled 8
  withcolor \mpcolor{spotA}
  withoutcolor ;
```

or preferably,

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

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

Besides the `mplibcode` environment (for  $\text{\LaTeX}$ ) and `\mplibcode ... \endmplibcode` (for Plain), we also provide unexpandable  $\text{\TeX}$  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
```



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

### 1.1.15 About cache files

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

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

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

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

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

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

### 1.1.16 About figure box metric

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

### 1.1.17 `luamplib.cfg`

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

### 1.1.18 Tagged PDF

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

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

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

**actualtext=**`<text>` starts a Span tag implicitly and sets a replacement text (a.k.a. actual text) from the `<text>`. If in vertical mode, horizontal mode will be forced by `\noindent` command.<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 T<sub>E</sub>X-text boxes (such as `btex ... etex`, excluding pictures made by `infont` operator). If in vertical mode, horizontal mode will be 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 T<sub>E</sub>X-text boxes in the order they are drawn in the figure.

N.B. Within text-mode figures, reusing T<sub>E</sub>X-text boxes is strongly discouraged.

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

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

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

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

**tag=**`<name>` You can choose a tag name, default value being Figure.<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.

<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.

<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 L<sup>A</sup>T<sub>E</sub>X's regular math formula tagging, for which the `text` key is needed.

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

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

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

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

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

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

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

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

## 1.2 METAPOST

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

`mplibdimen  $\langle$ string $\rangle$`  and `mplibcolor  $\langle$ string $\rangle$`  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  $\langle$ string $\rangle$`  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";
```

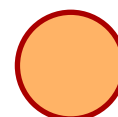
But the result may vary in its color model (gray/rgb/cmyk) according to the given  $\TeX$  color. Therefore the example shown above would raise a `METAPOST error: cmykcolor col; should have been declared`. By contrast, `mplibrbgtexcolor`  $\langle string \rangle$  always returns rgb-model expressions.

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

### 1.2.3 `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`  $(\langle fill\ color\ expr \rangle, \langle stroke\ color\ expr \rangle)$ . When the argument is in string type, it is regarded as the color expression of  $\TeX$  side. A simple example (see also the example at § 1.2.8):

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

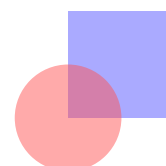


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

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

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

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



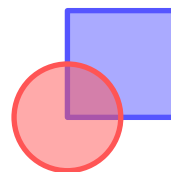
---

<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.

### 1.2.5 withmplibopacities (... , ... , ...)

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

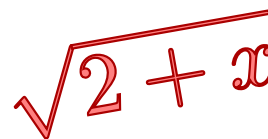
```
\mpfig
pickup pencircle scaled 2;
filldraw unitsquare scaled 40
  withcolor 1/3[blue,white]
  withmplibopacities (1, 1/2, 1)    % or ("normal", 1/2, 1)
;
filldraw fullcircle scaled 40
  withcolor 1/3[red,white]
  withmplibopacities (1, 1/2, 1)
;
\endmpfig
```



### 1.2.6 mplibgraphicstext ...

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

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



`fakebold`, `fillcolor` and `drawcolor` (or `strokecolor`) are optional; default values are 2, "white" and "black" respectively.<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.

<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 below § 1.2.12.

### 1.2.7 `mplibglyph` ... of ...

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

```
mplibglyph 50 of \fontid\font          % slot 50 of current font
mplibglyph "Q" of "TU/TeXGyrePagella(0)/m/n/10" % font csname
mplibglyph "Q" of "texgyrepagella-regular.otf" % raw filename
mplibglyph "R" of "utmr8a.pfb" % raw filename (type1 font)
mplibglyph "Q" of "Times.ttc(2)" % subfont number
mplibglyph "Q" of "SourceHanSansK-VF.otf[Regular]" % instance name
mplibglyph "R" of "SourceHanSansK-VF.otf[wght=800]" % axis names & values
```

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

N.B. Regrettably we have some bug in processing not a few glyphs in `cmr10.pfb` and its family (or maybe other) Type1 fonts.<sup>12</sup> If that happens, consider using glyph operator instead of `mplibglyph`.

### 1.2.8 `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`  $\langle picture \rangle$  command fills the paths according to the nonzero winding number rule. As a result, for instance, the area surrounded by inner path of “O” will remain transparent.

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

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

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

<sup>12</sup>The bug seems to be fixed in `font-cff.lmt` contained in Con $\TeX$ t mkxl, but current `luaotfload` is based on `font-cff.lua` from Con $\TeX$ t mkiv. As you see, `mplibglyph` operator requires `luaotfload` package loaded, which however is done automatically by  $\mathcal{E}\mathcal{T}\mathcal{E}\mathcal{X}$  format.

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

An example:

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



### 1.2.9 mpliboutlinetext (...)

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

A simple example:

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



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

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

### 1.2.10 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 installed.

### 1.2.11 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 installed.

### 1.2.12 ... 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* here means a picture generated by `btex ... etex`, `texttext`, `TEX`, `maketext`, `mplibgraphictext` (see below § 1.2.6), or `infont` operator, though technically only the last one is a true textual picture. Note that the picture, including transparency group, in which the objects are filled *without* color can also be regarded as a textual picture.<sup>14</sup>

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



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

The syntax is `<path> | <textual picture> withshadingmethod <string>`, where the latter shall be "linear", "circular", "triangle", "lattice", "coons", or "tensor". The balance of this subsection is to explain additional optional macros. Above all, there are two ways in specifying the shading coordinates, of which you can choose the more convenient one. First, the way that mimicks the *metafun*'s:

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

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

---

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

**withshadingorigin**  $\langle pair \rangle$  The center of both starting and ending circles, default value being center p, where p is the operand of withshadingmethod.

**withshadingcenter**  $\langle pair \rangle$  Value to specify the starting center. For instance, (0,0) means that the center of starting circle is center p; (1,1) means urcorner p; (-1,-1) means llcorner p.

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

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

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

Secondly, the way provided by luamplib only:

**withshadingpoints** ( $\langle pair \rangle$ ,  $\langle pair \rangle$ ) In linear mode, values to specify directly the starting and ending points: you can use it instead of withshadingvector or withshadingdirection. In circular mode, the centers of starting and ending circles: it could be easier than issuing two macros withshadingorigin and withshadingcenter. Note that, within the macro, both withshadingfactor 1 and withshadingtransform "no" are already decalred.

**withshadingcenters** ( $\langle pair \rangle$ ,  $\langle pair \rangle$ ) Synonym of withshadingpoints. Normally accompanied by withshadingradius which has the same meaning as described above.

Now, optional macros common to the both ways:

**withshadingstep** (...) For combined shading of more than two colors.

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

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

**withshadingdomain**  $\langle pair \rangle$  Limiting values of parametric variable that varies on the axis of color gradient, default value being (0,1). Of course the values can be negative or greater than 1.

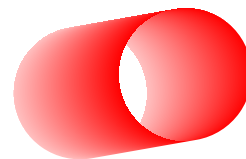
**withshadingextend** ( $\langle boolean \rangle$ ,  $\langle boolean \rangle$ ) Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

```
\mpfig
path p[];
p1 = fullcircle scaled 50;
p2 = fullcircle scaled 50 shifted 40 right;
```

```

fill (subpath (2,6) of p1 -- subpath (-2,2) of p2 -- cycle) rotated 10
  withshadingmethod "circular"
  withshadingcenters (center p1, center p2 rotated 10)
  withshadingradius (25, 25)
  withshadingcolors (3/4[red,white], red)
  withshadingextend (false, false) ;
\endmpfig

```



**withshadingmatrix**  $\langle \text{string} \rangle$  METAPOST code for transformation of shading, such as "xscaled 1.2 yscaled 0.8"; or six numerics separated by spaces, such as "1.2 0 0 0.8 0 0".

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

### 1.2.13 ... withfademethod ...

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

Related macros to control optional values are:

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

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

**withfadecenter** is a synonym of withfadevector.

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

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

**withfadefraction**  $\langle \text{numeric} \rangle$  Fractional number of each fading step. Only meaningful within withfadestep.

**withfadeextend** ( $\langle \text{boolean} \rangle$ ,  $\langle \text{boolean} \rangle$ ) specifies whether to extend the fading beyond the starting and ending points or circles, default value being (true, true).

**withfadematrix**  $\langle string \rangle$  METAPOST code for transformation of fading, such as "xscaled 1.2 yscaled 0.8"; or six numerics separated by spaces, such as "1.2 0 0 0.8 0 0".

**withfadebbox** ( $\langle pair \rangle$ ,  $\langle pair \rangle$ ) 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 § 1.2.15 on the analogous macro withgroupbbox.

An example:

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



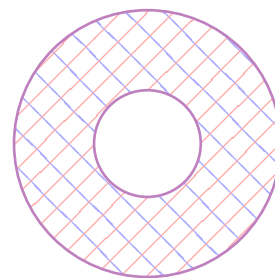
#### 1.2.14 \mppattern{...} ... \endmppattern, ... withmppattern ...

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

An example:

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

\mpfig
  mplibdrawglyph image(
    draw fullcircle scaled 100;
    draw reverse fullcircle scaled 40;
  )
```



<sup>15</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,  $\langle cyclic path \rangle$  withmppattern  $\langle string \rangle$  works as intended only with fill or filldraw command.

Table 1: options for \mppattern

Key	Value Type	Explanation
xstep	<i>number</i>	horizontal spacing between pattern cells
ystep	<i>number</i>	vertical spacing between pattern cells
xshift	<i>number</i>	horizontal shifting of pattern cells
yshift	<i>number</i>	vertical shifting of pattern cells
bbox	<i>table or string</i>	llx, lly, urx, ury values*
matrix	<i>table or 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

```

withmppattern "mypatt"
withpen pencircle scaled 1
withcolor \mpcolor{red!50!blue!50} ;
\endmpfig

```

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

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

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

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

```

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

\begin{mplibcode}
beginfig(1)
  picture tex;
  tex = mpliboutlinetext ("bfseries \TeX");

```

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

```

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

```



#### 1.2.15 ... asgroup ...

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

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

- As mentioned, in addition to those arguments mimicking *metafun*'s, we allow other optional arguments at the right-hand side:
  - `asgroup "off"` will produce an ordinary *form XObject* rather than a transparency group XObject. By contrast, a transparency group will be produced when 'off' is not given, including `""` (empty string) in which case both of the PDF keys `/I` and `/K` are false.
  - `asgroup "wrapped"` will produce a transparency group XObject in which an ordinary form XObject is wrapped up. This option could be useful when a picture shading

(*shading pattern* in technical terms) is used in the object of `asgroup`. See the note about picture shading described [below](#).

- You can reuse the XObject as many times as you want in the  $\TeX$  code or in other `METAPOST` code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide  $\TeX$  and `METAPOST` macros as follows:

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

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

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

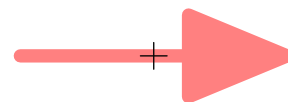
**withgroupbbox** ( $\langle pair \rangle$ ,  $\langle pair \rangle$ ) sets the bounding box of the XObject, default value being (llcorner p, urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence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

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

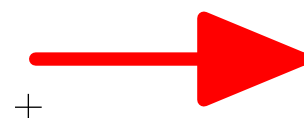
```
\mpfig
picture pic;
pic = image(drawarrow (left--right) scaled 5 withcolor red) scaled 10 ;
draw pic
  asgroup "off"
  withtransparency (1, 1/2) ;
\endmpfig
```



```
\mpfig
draw pic
  asgroup ""
  withgroupname "mygroup"
  withtransparency (1, 1/2) ;
draw (left--right) scaled 5 ;
draw (up--down) scaled 5 ;
\endmpfig
```



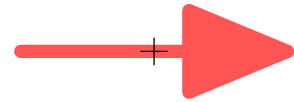
```
\noindent
\clap{\vrule width 10bp height .25bp depth .25bp}%
\clap{\vrule width .5bp height 5bp depth 5bp}%
\usemplibgroup{mygroup}
```



```

\mpfig
  usemplibgroup "mygroup"
    withtransparency (1, 2/3) ;
    draw (left--right) scaled 5 ;
    draw (up--down) scaled 5 ;
\endmpfig

```



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

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

```

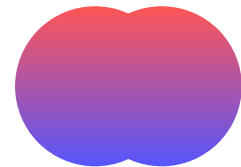
\mpfig*
  picture pic[];
  pic1 = image(
    fill fullcircle scaled 60 withoutcolor;
    fill fullcircle scaled 60 shifted 25right withoutcolor;
  ) ;
  pic2 = image(
    draw pic1
      withshadingmethod "linear"
      withshadingvector (2, 1)
      withshadingcolors (red, blue)
  ) ;
\endmpfig

```

```

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

```



```

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

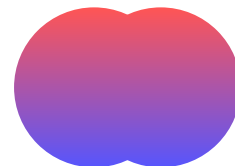
```



After some experiment, however, it turned out that the best way to get picture shading with transparency group is to give shading effect within an ordinary form XObject and then wrap it up in a transparency group XObject. This sort of double wrapping will be done automatically

when you specify ‘wrapped’ as an optional argument of `asgroup`. Most PDF renderers do render it properly:

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



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

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

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



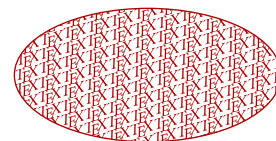
### 1.2.16 `\mplibgroup{...} ... \endmplibgroup`

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

An example:<sup>17</sup>

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

\usemplibgroup{texpatt}
```




---

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

Table 2: options for \mplibgroup

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

\* in string type, numbers are separated by spaces

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



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

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

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

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

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

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

### 1.2.17 ... withmaskinggroup ...

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

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

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

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

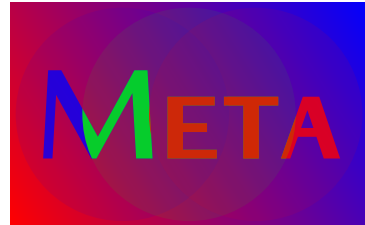
**withmaskingbgcolor**  $\langle$ *color expr* $\rangle$  sets the background color of the masking group. 0 denotes full transparency (masking invisibly); 1, full color.<sup>20</sup>

An example:

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

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

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



### 1.2.18 Free-form Gouraud-shaded triangle mesh shading

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

Optional macros for triangle mesh shading:

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

<sup>20</sup>Color expressions in *rgb* or *cmyk* model are also allowed, in accordance with the option *colorspace*="/DeviceRGB" or *colorspace*="/DeviceCMYK" given to the masking transparency group. This however we do not recommend as the luminosity is difficult to understand intuitively.

**withtrianglepatchinit** ( $\langle path \rangle$  |  $\langle string \rangle$ ,  $\langle color \rangle$ ,  $\langle color \rangle$ ,  $\langle color \rangle$ ) The initial triangle.

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

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

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

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

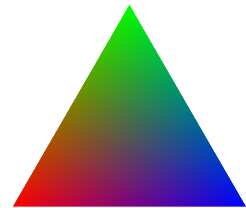
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space ( $x$  and  $y$  coordinates).

The third argument is the color for the new vertex.

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

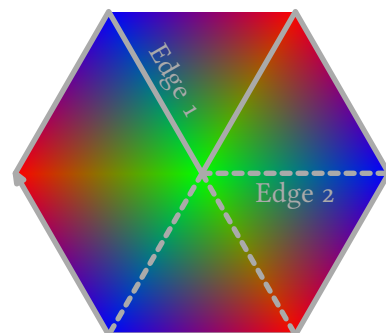
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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



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

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



### 1.2.19 Lattice-form Gouraud-shaded triangle mesh shading

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

Optional macros for lattice-form shading:

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

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

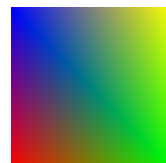
The first argument shall be a pair, or a string of two numerics separated by a space ( $x$  and  $y$  coordinates). The second argument shall be a color expression for the vertex.

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

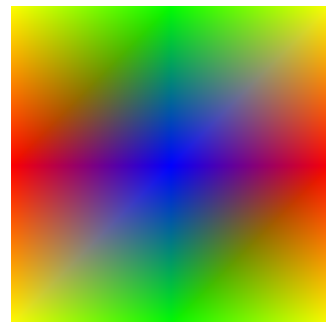
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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

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



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



### 1.2.20 Coons patch mesh shading

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

Optional macros for Coons patch mesh shading:

**withcoonspatchinit** ( $\langle path \rangle$  |  $\langle string \rangle$ ,  $\langle color \rangle$ ,  $\langle color \rangle$ ,  $\langle color \rangle$ ,  $\langle color \rangle$ ) The initial patch.

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

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

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

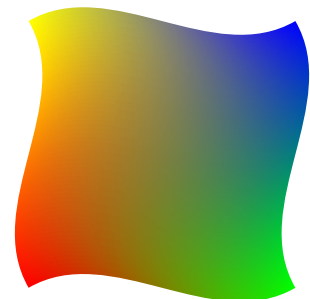
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

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

Remaining arguments are two color expressions for the new vertices.

Examples showing the Coons patch shading and illustrating the usage of the optional macros:<sup>21</sup>

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



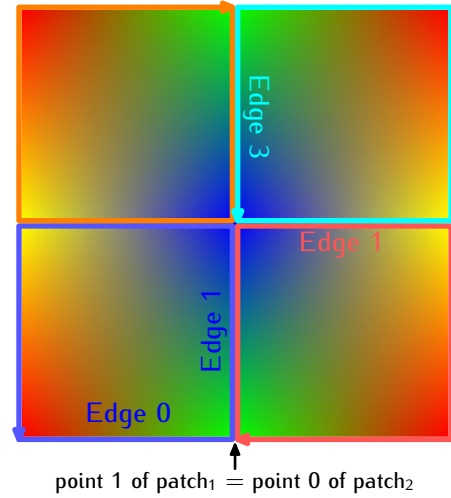
```
\mpfig
u := 80 ;
path p; p = unitsquare scaled u;
def fsquare (expr n) =
  ( point n of p --
    point n+1 of p --
    point n+2 of p --
    point n+3 of p -- cycle )
enddef;
```

<sup>21</sup>Be aware that currently Mac OS Preview exposes its some bug in rendering the second figure below, especially when the edge flag is 3.

```

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

```



### 1.2.21 Tensor-product patch mesh shading

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

Optional macros for tensor-product patch mesh shading:

**withtensorpatchinit** ( $\langle path \rangle | \langle string \rangle, \langle path \rangle | \langle string \rangle, \langle color \rangle, \langle color \rangle, \langle color \rangle, \langle color \rangle$ ) The initial patch.

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

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

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

An example:<sup>22</sup>

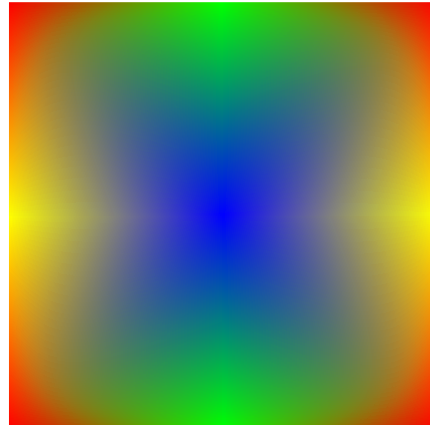
```
\mpfig
```

<sup>22</sup>Be aware that currently Mac OS Preview exposes its some bug in rendering the figure below.

```

fill p scaled 2
  withshadingmethod "tensor"
  withtensorpatchinit (
    fsquare(0),
    fsquare(0) scaled 0.1 shifted (0.1u, 0.1u),
    red, green, blue, yellow)
  withtensorpatchnext (1,
    rsquare(0) shifted (u,0),
    rsquare(0) scaled 0.1 shifted (1.8u, 0.1u),
    yellow, red)
  withtensorpatchnext (1,
    fsquare(0) shifted (u,u),
    fsquare(0) scaled 0.1 shifted (1.8u, 1.8u),
    red, green)
  withtensorpatchnext (3,
    rsquare(2) shifted (0,u),
    rsquare(2) scaled 0.1 shifted (0.1u, 1.8u),
    yellow, red) ;
\endmpfig

```



## 1.3 Lua

### 1.3.1 runscript ...

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

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

### 1.3.2 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}

```

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
legacyverbatimex	<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

```

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

```

Of course, this sort of Lua code can also be run inside METAPOST code using `runscript` command. Again, of course you can access a METAPOST variable using your own T<sub>E</sub>X 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 § 1.2.14.

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

The first argument is the register of a box containing a pattern cell, which should be prepared in advance by the user. For instance, `\setbox0=\hbox{\tiny\TeX}`, or corresponding Lua code using `tex.setbox` function; then the argument should be 0.<sup>23</sup>

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 § 1.2.16.

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

The first argument is the register of a box prepared in advance by the user. When the contents of the box have been generated from  $\TeX$  (not  $\text{METAPOST}$ ) code, please make sure that both of the  $\TeX$  macros ‘`MPllx`’ and ‘`MPlly`’ are defined as ‘`0`’ before invoking the Lua function.<sup>24</sup>

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

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

## 2 Implementation

### 2.1 Lua module

```
1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.42.2",
5   date      = "2026/06/19",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8
```

Use the `luamplib` namespace, since `mplib` is for the  $\text{METAPOST}$  library itself.  $\text{Con}\TeX\text{t}$  uses `metapost`.

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

---

<sup>23</sup>In DVI mode,  $\TeX$  macro ‘`mplibpatternname`’ should be set as *<pattern name>* before preparing the box, if shading pattern (i.e. shading on picture) is used in the pattern cell.

<sup>24</sup>In DVI mode,  $\TeX$  macro ‘`mplibgroupname`’ also should be set as *<group name>* before preparing the box, if shading pattern (i.e. shading on picture) is used in the `mplibgroup`.

```

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

```

Provide a few “shortcuts” expected by the code.

```

50 local tableconcat = table.concat
51 local tableinsert = table.insert
52 local tableunpack = table.unpack
53 local texsprint   = tex.sprint
54 local texgettoks  = tex.gettoks
55 local texgetbox   = tex.getbox
56 local texruntoks  = tex.runtoks

```

```

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

```

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

```

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

```

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

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

```

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

```

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

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

```

Pattern formats to replace btex and verbatimtex ... 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 verbatimetex_etex = name_b.."verbatimetex"..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(verbatimtex_etex, "verbatimtex %1 etex;") -- semicolon
192 count = count + cnt
193 if count == 0 then
194     noneedtoreplace[name] = true
195     fh = ioopen(newfile,"w");
196     if fh then
197         fh:close()
198         lfstouch(newfile,currenttime,ofmodify)
199     end
200     return file
201 end
202 fh = ioopen(newfile,"w")
203 if not fh then return file end
204 fh:write(data); fh:close()
205 lfstouch(newfile,currenttime,ofmodify)
206 return newfile
207 end

```

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

```

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

```

```

237     kpse.record_input_file(file) -- recorder
238     end
239     return file
240   end
241 end
242 end
243

```

For the main function: process

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

```

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

```

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

```

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

```

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

```

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

```

v2.6.1: now luamplib does not disregard show command, even when luamplib.showlog is false.

Incidentally, it does not raise error nor prints an info, even if output has no figure.

```

275     local show = log:match"\n>>? .+"
276     if show then
277       termorlog("term", show, "Info (more info in the log)")

```

```

278     info(log)
279     elseif luamplib.showlog and log:find"%g" then
280         info(log)
281     end
282 end
283 return log
284 end
285 end

```

lualibs-os.lua installs a randomseed. When this file is 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 local process_stack = 0
315 function process (data, instancename)
316     local currfmt
317     process_stack = process_stack + 1
318     if instancename and instancename ~= "" then

```

```

319     currfmt = instancename
320     has_instancename = true
321 else
322     currfmt = tableconcat{
323         currentformat,
324         luaplib.numbersystem or "scaled",
325         tostring(luaplib.texttextlabel),
326         tostring(luaplib.legacyverbatimtex),
327         tostring(process_stack), -- try to address #63
328     }
329     has_instancename = false
330 end
331 local mpx = mplibinstances[currfmt]
332 local standalone = not (has_instancename or luaplib.codeinherit)
333 if mpx and standalone then
334     mpx:finish()
335 end
336 local log = ""
337 if standalone or not mpx then
338     mpx, _, log = luaplibload(currentformat)
339     mplibinstances[currfmt] = mpx
340 end
341 local converted, result = false, {}
342 if mpx and data then
343     result = mpx:execute(data)
344     local log = reporterror(result, log)
345     if log then
346         if result.fig then
347             converted = luaplib.convert(result)
348         end
349     end
350 else
351     err"Mem file unloadable. Maybe generated with a different version of mplib?"
352 end
353 process_stack = process_stack - 1
354 return converted, result
355 end
356 end
357

```

dvipdfmx is supported, though nobody seems to use it.

```

358 local pdfmode = tex.outputmode > 0
359

```

make\_text and some run\_script uses Lua<sub>T<sub>E</sub>X</sub>'s tex.runtoks.

```

360 local catlatex = luatexbase.registernumber("catcodetable@latex")
361 local catat11 = luatexbase.registernumber("catcodetable@atletter")

```

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

```

362 local function run_tex_code (str, cat)

```

```

363 texruntoks(function() texsprint(cat or catlatex, str) end)
364 end

```

For conversion of sp to bp.

```

365 local factor = 65536*(7227/7200)
366

```

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.

```

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

```

405

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

```

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

```

A common function for color functions

```

420 local function colorsplit (res)
421   local t, tt = { }, res:gsub("[%%]", "", 2):explode()
422   local be = tt[1]:find"^%" and 1 or 2
423   for i=be, #tt do
424     if not tonumber(tt[i]) then break end
425     t[#t+1] = tt[i]
426   end
427   if #t == 0 then -- named color in DVI mode with no DocumentMetadata
428     run_tex_code{"\extractcolorspecs", tt[3], "\mplibtmpa\mplibtmpb"}
429     t = get_macro"mplibtmpb":explode",
430   end
431   return t
432 end
433 do
434   local colfmt = ccexplat and "l3color" or "xcolor"
435   local mplibcolorfmt = {
436     xcolor = tableconcat{
437       [[\begingroup\let\XC@mcolor\relax]],
438       [[\def\set@color{\global\mplibtmp toks\expandafter{\current@color}}]],
439       [[\color%s\endgroup]],
440     },
441     l3color = tableconcat{
442       [[\begingroup\def\__color_select:N#1{\expandafter\__color_select:nn#1}]],
443       [[\def\__color_backend_select:nn#1#2{\global\mplibtmp toks{#1 #2}}]],
444       [[\def\__kernel_backend_literal:e#1{\global\mplibtmp toks\expandafter{\expanded{#1}}}],
445       [[\color_select:n%s\endgroup]],
446     },
447   }
448   function process_color (str)
449     if str then

```

```

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

```

498

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

```

499 local function process_verbatimtex_text (str)
500   if str then
501     run_tex_code(str)
502   end
503   return ""
504 end
505
```

For legacy verbatimtex process. verbatimtex ... etex before beginfig() is inserted just before the mplib box. And T<sub>E</sub>X code inside beginfig() ... endfig is inserted after the mplib box.

```

506 local tex_code_pre_mplib = {}
507 luamplib.figid = 1
508 luamplib.in_the_fig = false
509 local function process_verbatimtex_prefig (str)
510   if str then
511     tex_code_pre_mplib[luamplib.figid] = str
512   end
513   return ""
514 end
515 local function process_verbatimtex_infig (str)
516   if str then
517     return format('special "postmplibverbtex=%s";', str)
518   end
519   return ""
520 end
521
```

For *metafun* format. see issue #79.

```

522 mp = mp or {}
523 local mp = mp
524 mp.mf_path_reset = mp.mf_path_reset or function() end
525 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
526 mp.report = mp.report or info

```

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

```

527 catcodes = catcodes or {}
528 local catcodes = catcodes
529 catcodes.numbers = catcodes.numbers or {}
530 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
531 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
532 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
533 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
534 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
535 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
536 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
537
```

Now luamplib.runscript

```

538 do
539   local runscript_funcs = {
540     luamplibtext    = process_tex_text,
541     luamplibcolor   = process_mplibcolor,
542     luamplibdimen   = process_dimen,
543     luamplibprefig  = process_verbatimtex_prefig,
544     luamplibinfig   = process_verbatimtex_infig,
545     luamplibverbtex = process_verbatimtex_text,
546   }

```

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

```

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

```

```

586     return tableconcat(buffer)
587 end
588 return ""
589 end
590 end
591
    luamplib.maketext
592 luamplib.legacyverbatimex = true
593 do
make_text must be one liner, so comment sign is not allowed.
594 local function protecttexcontents (str)
595     return str:gsub("\\%", "\\0PerCent\0")
596           :gsub("%%.-\n", "")
597           :gsub("%%.-$", "")
598           :gsub("%zPerCentz", "\\%")
599           :gsub("\r.-$", "")
600           :gsub("%s+", " ")
601 end
602 function luamplib.maketext (str, what)
603     if str and str ~= "" then
604         str = protecttexcontents(str)
605         if what == 1 then
606             if not str:find("\\documentclass"..name_e) and
607                not str:find("\\begin{s*{document}") and
608                not str:find("\\documentstyle"..name_e) and
609                not str:find("\\usepackage"..name_e) then
610                 if luamplib.legacyverbatimex then
611                     if luamplib.in_the_fig then
612                         return process_verbatimex_infig(str)
613                     else
614                         return process_verbatimex_prefig(str)
615                     end
616                 else
617                     return process_verbatimex_text(str)
618                 end
619             end
620         else
621             return process_tex_text(str, true) -- bool is for 'char13'
622         end
623     end
624     return ""
625 end
626 end
627
    luamplib's METAPOST color operators
628 luamplib.gettexcolor = function (str, rgb)
629     local res = process_color(str):match'"mpliboverridecolor=(.+)"'
630     if res:find" cs " or res:find"@pdf.obj" then

```

```

631   if not rgb then
632     warn("%s is a spot color. Forced to CMYK", str)
633   end
634   run_tex_code({
635     "\\color_export:nnN{" ,
636     str,
637     "}" ,
638     rgb and "space-sep-rgb" or "space-sep-cmyk",
639     "}"\mplib@tempa",
640   },ccexplat)
641   return get_macro"mplib@tempa":explode()
642 end
643 local t = colorsplit(res)
644 if #t == 3 or not rgb then return t end
645 if #t == 4 then
646   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]) }
647 end
648 return { t[1], t[1], t[1] }
649 end
650
651 luamplib.shadecolor = function (str)
652   local res = process_color(str):match'"mpliboverridecolor=(.)"'
653   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}

654   run_tex_code({
655     [[\color_export:nnN{]], str, [[]{backend}\mplib@tempa]],
656   },ccexplat)
657   local name, value = get_macro'mplib@tempa':match'{{(.-)}}{(.-)}'
658   local t, obj = res:explode()
659   if pdfmode then
660     obj = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
661   else
662     obj = t[2]
663   end
664   return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
665 end
666 return colorsplit(res)
667 end
668

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

```

```

700   return format('withprescript "mpliboverridecolor=%s%s %s"', bc, fill, stroke)
701 end
702 end
703

```

Remove trailing zeros for smaller PDF

```

704 local decimals = "%.d+"
705 local function rmzeros(str) return str:gsub("%.?0+$", "") end
706

```

common function for mplibgraphicstext and mpliboutlinetext

```

707 local function getrulemetric (box, curr, bp)
708   local running = -1073741824
709   local wd,ht,dp = curr.width, curr.height, curr.depth
710   wd = wd == running and box.width or wd
711   ht = ht == running and box.height or ht
712   dp = dp == running and box.depth or dp
713   if bp then
714     return wd/factor, ht/factor, dp/factor
715   end
716   return wd, ht, dp
717 end
718

```

luamplib's mplibgraphicstext operator

```

719 do
720   if not math.round then
721     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
722   end
723   local emboldenfonts = { }
724   local function roundupwidth (f, fb)
725     local wd = math.round(f.size * fb / factor * 10)
726     if wd == 0 and fb ~= 0 then
727       wd = 1
728     end
729     emboldenfonts.width = wd
730     return wd
731   end
732   local function getemboldenwidth (curr, fakebold)
733     local width = emboldenfonts.width
734     if not width then
735       local f
736       local function getglyph(n)
737         while n do
738           if n.head then
739             getglyph(n.head)
740           elseif n.font and n.font > 0 then
741             f = n.font; break
742           end
743           n = node.getnext(n)

```

```

744     end
745   end
746   getglyph(curr)
747   width = roundupwidth(font.getcopy(f or font.current()), fakebold)
748   end
749   return width
750 end
751 local function getrulewhatsit (line, wd, ht, dp)
752   line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
753   line = line == 0 and "" or ("%f w"):format(line)
754   local pl
755   local fmt = "q %s %f %f %f %f re B Q"
756   if pdfmode then
757     pl = node.new("whatsit", "pdf_literal")
758     pl.mode = 0
759   else
760     fmt = "pdf:content " .. fmt
761     pl = node.new("whatsit", "special")
762   end
763   pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
764   local ss = node.new"glue"
765   node.setglue(ss, 0, 65536, 65536, 2, 2)
766   pl.next = ss
767   return pl
768 end

```

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

```

769 local tag_update_attrs
770 if is_defined"ver@tagpdf.sty" then
771   tag_update_attrs = function (n, curr)
772     while n do
773       n.attr = curr.attr
774       if n.head then
775         tag_update_attrs(n.head, curr)
776       end
777       n = node.getnext(n)
778     end
779   end
780 else
781   tag_update_attrs = function() end
782 end
783 local function embolden (box, curr, fakebold)
784   local head = curr
785   while curr do
786     if curr.head then
787       curr.head = embolden(curr, curr.head, fakebold)
788     elseif curr.replace then
789       curr.replace = embolden(box, curr.replace, fakebold)
790     elseif curr.leader then

```

```

791     if curr.leader.head then
792         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
793     elseif curr.leader.id == node.id"rule" then
794         local glue = node.effective_glue(curr, box)
795         local line = getemboldenwidth(curr, fakebold)
796         local wd,ht,dp = getrulemetric(box, curr.leader)
797         if box.id == node.id"hlist" then
798             wd = glue
799         else
800             ht, dp = 0, glue
801         end
802         local pl = getrulewhatsit(line, wd, ht, dp)
803         local pack = box.id == node.id"hlist" and node.hpack or node.vpack
804         local list = pack(pl, glue, "exactly")
805         tag_update_attrs(list,curr)
806         head = node.insert_after(head, curr, list)
807         head, curr = node.remove(head, curr)
808     end
809 elseif curr.id == node.id"rule" and curr.subtype == 0 then
810     local line = getemboldenwidth(curr, fakebold)
811     local wd,ht,dp = getrulemetric(box, curr)
812     if box.id == node.id"vlist" then
813         ht, dp = 0, ht+dp
814     end
815     local pl = getrulewhatsit(line, wd, ht, dp)
816     local list
817     if box.id == node.id"hlist" then
818         list = node.hpack(pl, wd, "exactly")
819     else
820         list = node.vpack(pl, ht+dp, "exactly")
821     end
822     tag_update_attrs(list,curr)
823     head = node.insert_after(head, curr, list)
824     head, curr = node.remove(head, curr)
825 elseif curr.id == node.id"glyph" and curr.font > 0 then
826     local f = curr.font
827     local key = format("%s:%s",f,fakebold)
828     local i = emboldenfonts[key]
829     if not i then
830         local ft = font.getfont(f) or font.getcopy(f)
831         local width = roundupwidth(ft, fakebold)
832         if ft.format == "opentype" or ft.format == "truetype" then
833             local name = ft.name:gsub("'",''):gsub('$','')
834             local t = name:gsub("^file:",''):gsub("^name:",''):gsub("^kpse:",''):gsub("^my:",'')
835             name = format('%s%sembolden=%s;',name, t:find":" and ";" or ":", fakebold)
836             _, i = fonts.constructors.readanddefine(name,ft.size)
837         elseif pdfmode then
838             local ft = table.copy(ft)
839             ft.mode, ft.width = 2, width

```

```

840         i = font.define(ft)
841     else
842         goto skip_type1
843     end
844     emboldenfonts[key] = i
845     end
846     curr.font = i
847 end
848 ::skip_type1::
849     curr = node.getnext(curr)
850 end
851 return head
852 end
853 luamplib.graphicstext = function (text, fakebold, fc, dc)
854     local fmt = process_tex_text(text):sub(1,-2)
855     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
856     emboldenfonts.width = nil
857     local box = texgetbox(id)
858     box.head = embolden(box, box.head, fakebold)
859     local colors = luamplib.fillandstrokecolor(fc, dc)
860     return format('%s %s)', fmt, colors)
861 end
862 end
863

```

#### luamplib's mplibglyph operator

```

864 do
865     local function mperr (str)
866         return format("hide(errmessage %q)", str)
867     end
868     local function getangle (a,b,c)
869         local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
870         if r > 180 then
871             r = r - 360
872         elseif r < -180 then
873             r = r + 360
874         end
875         return r
876     end
877     local function turning (t)
878         local r, n = 0, #t
879         for i=1,2 do
880             tableinsert(t, t[i])
881         end
882         for i=1,n do
883             r = r + getangle(t[i], t[i+1], t[i+2])
884         end
885         return r/360
886     end

```

```

887 local function glyphimage(t, fmt)
888     local q, p, r, towarn = {}, {}
889     local function closepath(dots)
890         tableinsert(p, format("%scycle", dots or "--"))
891         tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
892     end
893     for i,v in ipairs(t) do
894         local cmd = v[#v]
895         local nt = t[i+1]
896         local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
897         if cmd == "m" then
898             if final then towarn = true end
899             p = {format('(%s,%s)',v[1],v[2])}
900             r = {{x=v[1],y=v[2]}}
901         else
902             if cmd == "l" then
903                 local pt = t[i-1]
904                 if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
905                     else
906                         tableinsert(p, format('--(%s,%s)',v[1],v[2]))
907                         tableinsert(r, {x=v[1],y=v[2]})
908                     end
909                 if final then closepath() end
910             elseif cmd == "c" then
911                 tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
912                 if final and r[1].x == v[5] and r[1].y == v[6] then
913                     closepath ".."
914                 else
915                     tableinsert(p, format('..(%s,%s)',v[5],v[6]))
916                     tableinsert(r, {x=v[5],y=v[6]})
917                     if final then closepath() end
918                 end
919             elseif cmd == "path" or cmd == "move" then
920                 else
921                     return mperr"unknown operator"
922                 end
923             end
924         end
925         r = { }
926         if fmt == "opentype" then
927             for _,v in ipairs(q[1]) do
928                 tableinsert(r, format('addto currentpicture contour %s;',v))
929             end
930             for _,v in ipairs(q[2]) do
931                 tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
932             end
933         else
934             for _,v in ipairs(q[2]) do
935                 tableinsert(r, format('addto currentpicture contour %s;',v))

```

```

936     end
937     for _,v in ipairs(q[1]) do
938         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
939     end
940 end
941 return format('image(%s)', tableconcat(r)), towarn
942 end
943 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
944 function luamplib.glyph (f, c)
945     local filename, subfont, instance, kind, shapedata
946     local fid = tonumber(f) or font.id(f)
947     if fid > 0 then
948         local fontdata = font.getfont(fid) or font.getcopy(fid)
949         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
950         instance = fontdata.specification and fontdata.specification.instance
951             or fontdata.shared and fontdata.shared.features.axis
952         filename = filename and filename:gsub("^harfloaded:", "")
953     else
954         local name
955         f = f:match"^%s*(.)%s*$"
956         name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]%"
957         if not name then
958             name, instance = f:match"(.+)%[(.-)]%" -- SourceHanSansK-VF.otf[Heavy]
959         end
960         if not name then
961             name, subfont = f:match"(.+)%((%d+)%)$" -- Times.ttc(2)
962         end
963         name = name or f
964         subfont = (subfont or 0)+1
965         instance = instance and instance:lower()
966         for _,ftype in ipairs{"opentype", "truetype"} do
967             filename = kpse.find_file(name, ftype.." fonts")
968             if filename then
969                 kind = ftype; break
970             end
971         end
972     end
973     if kind ~= "opentype" and kind ~= "truetype" then
974         f = fid and fid > 0 and tex.fontname(fid) or f
975         if kpse.find_file(f, "tfm") then
976             return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
977         else
978             filename = kpse.find_file(f, "type1 fonts")
979             if filename then
980                 kind = "type1" -- there's bug in processing cmr family
981             else
982                 return mperr"font not found"
983             end
984         end
985     end

```

```

985     end
986     local time = lfsattributes(filename,"modification")

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

987     local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
988     local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
989     local newname = format("%s/%s.lua", cachedir or outputdir(), h)
990     local newtime = lfsattributes(newname,"modification") or 0
991     if time == newtime then
992         shapedata = require(newname)
993     end
994     if not shapedata then
995         if fonts then
996             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
997             shapedata = handler.readers.loadshapes(filename,subfont,instance)
998         end
999         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1000         table.tofile(newname, shapedata, "return")
1001         lfstouch(newname, time, time)
1002     end
1003     local gid = tonumber(c)
1004     if not gid then
1005         local uni = utf8.codepoint(c)
1006         for i,v in pairs(shapedata.glyphs) do
1007             if c == v.name or uni == v.unicode then
1008                 gid = i; break
1009             end
1010         end
1011     end
1012     if not gid then return mperr"cannot get GID (glyph id)" end
1013     local fac = 1000 / (shapedata.units or 1000)
1014     local t = shapedata.glyphs[gid]; t = t and t.segments
1015     if not t then return "image()" end
1016     for i,v in ipairs(t) do
1017         if type(v) == "table" then
1018             for ii,vv in ipairs(v) do
1019                 if type(vv) == "number" then
1020                     t[i][ii] = format("%.0f", vv * fac)
1021                 end
1022             end
1023         end
1024     end
1025     local result, towarn = glyphimage(t, shapedata.format or kind)
1026     if towarn then
1027         warn("mplibglyph %s not working properly. Use glyph instead", f)
1028     end
1029     return result

```

```

1030 end
1031 end
1032

```

mpliboutline<sub>text</sub> : based on mkiv's font-mps.lua

```

1033 do
1034   local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1035     unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1036   local outline_horz, outline_vert
1037   function outline_vert (res, box, curr, xshift, yshift)
1038     local b2u = box.dir == "LTL"
1039     local dy = (b2u and -box.depth or box.height)/factor
1040     local ody = dy
1041     while curr do
1042       if curr.id == node.id"rule" then
1043         local wd, ht, dp = getrulemetric(box, curr, true)
1044         local hd = ht + dp
1045         if hd ~= 0 then
1046           dy = dy + (b2u and dp or -ht)
1047           if wd ~= 0 and curr.subtype == 0 then
1048             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1049           end
1050           dy = dy + (b2u and ht or -dp)
1051         end
1052       elseif curr.id == node.id"glue" then
1053         local vwidth = node.effective_glue(curr,box)/factor
1054         if curr.leader then
1055           local curr, kind = curr.leader, curr.subtype
1056           if curr.id == node.id"rule" then
1057             local wd = getrulemetric(box, curr, true)
1058             if wd ~= 0 then
1059               local hd = vwidth
1060               local dy = dy + (b2u and 0 or -hd)
1061               if hd ~= 0 and curr.subtype == 0 then
1062                 res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1063               end
1064             end
1065           elseif curr.head then
1066             local hd = (curr.height + curr.depth)/factor
1067             if hd <= vwidth then
1068               local dy, n, iy = dy, 0, 0
1069               if kind == 100 or kind == 103 then -- todo: gleaders
1070                 local ady = abs(ody - dy)
1071                 local ndy = math.ceil(ady / hd) * hd
1072                 local diff = ndy - ady
1073                 n = math.floor((vwidth-diff) / hd)
1074                 dy = dy + (b2u and diff or -diff)
1075               else
1076                 n = math.floor(vwidth / hd)

```

```

1077         if kind == 101 then
1078             local side = vwidth % hd / 2
1079             dy = dy + (b2u and side or -side)
1080         elseif kind == 102 then
1081             iy = vwidth % hd / (n+1)
1082             dy = dy + (b2u and iy or -iy)
1083         end
1084     end
1085     dy = dy + (b2u and curr.depth or -curr.height)/factor
1086     hd = b2u and hd or -hd
1087     iy = b2u and iy or -iy
1088     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1089     for i=1,n do
1090         res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1091         dy = dy + hd + iy
1092     end
1093 end
1094 end
1095 end
1096 dy = dy + (b2u and vwidth or -vwidth)
1097 elseif curr.id == node.id"kern" then
1098     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1099 elseif curr.id == node.id"vlist" then
1100     dy = dy + (b2u and curr.depth or -curr.height)/factor
1101     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1102     dy = dy + (b2u and curr.height or -curr.depth)/factor
1103 elseif curr.id == node.id"hlist" then
1104     dy = dy + (b2u and curr.depth or -curr.height)/factor
1105     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1106     dy = dy + (b2u and curr.height or -curr.depth)/factor
1107 end
1108 curr = node.getnext(curr)
1109 end
1110 return res
1111 end
1112 function outline_horz (res, box, curr, xshift, yshift, discwd)
1113     local r2l = box.dir == "TRT"
1114     local dx = r2l and (discwd or box.width/factor) or 0
1115     local dirs = { { dir = r2l, dx = dx } }
1116     while curr do
1117         if curr.id == node.id"dir" then
1118             local sign, dir = curr.dir:match"(.)(...)"
1119             local level, newdir = curr.level, r2l
1120             if sign == "+" then
1121                 newdir = dir == "TRT"
1122                 if r2l ~= newdir then
1123                     local n = node.getnext(curr)
1124                     while n do
1125                         if n.id == node.id"dir" and n.level+1 == level then break end

```

```

1126         n = node.getnext(n)
1127     end
1128     n = n or node.tail(curr)
1129     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1130 end
1131 dirs[level] = { dir = r2l, dx = dx }
1132 else
1133     local level = level + 1
1134     newdir = dirs[level].dir
1135     if r2l ~= newdir then
1136         dx = dirs[level].dx
1137     end
1138 end
1139 r2l = newdir
1140 elseif curr.char and curr.font and curr.font > 0 then
1141     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1142     local gid = ft.characters[curr.char].index or curr.char
1143     local scale = ft.size / factor / 1000
1144     local slant = (ft.slant or 0)/1000
1145     local extend = (ft.extend or 1000)/1000
1146     local squeeze = (ft.squeeze or 1000)/1000
1147     local expand = 1 + (curr.expansion_factor or 0)/1000000
1148     local xscale, yscale = scale * extend * expand, scale * squeeze
1149     dx = dx - (r2l and curr.width/factor*expand or 0)
1150     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1151     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1152     local vertical = ""
1153     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1154         if ft.shared.features.vertical then -- luatexko
1155             vertical = "rotated 90"
1156             local data = ft.characters[curr.char] or { }
1157             if ft.hb then
1158                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1159                 local charraise = (ft.luatexko_charraise or 0)/factor
1160                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1161             else
1162                 local cmds = data.commands or { {0,0}, {0,0} }
1163                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1164                 xpos, ypos = xpos + hoff, ypos + voff
1165             end
1166         elseif curr ~= box.head then -- luatexja
1167             vertical = "rotated 90"
1168             local en = ft.parameters.quad/factor/2
1169             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1170         end
1171     end
1172     local image
1173     if ft.format == "opentype" or ft.format == "truetype" then
1174         image = luamplib.glyph(curr.font, gid)

```

```

1175     else
1176         local name, scale = ft.name, 1
1177         local vf = font.read_vf(name, ft.size)
1178         if vf and vf.characters[gid] then
1179             local cmds = vf.characters[gid].commands or {}
1180             for _,v in ipairs(cmds) do
1181                 if v[1] == "char" then
1182                     gid = v[2]
1183                 elseif v[1] == "font" and vf.fonts[v[2]] then
1184                     name = vf.fonts[v[2]].name
1185                     scale = vf.fonts[v[2]].size / ft.size
1186                 end
1187             end
1188         end
1189         image = format("glyph %s of %q scaled %f", gid, name, scale)
1190     end
1191     res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1192                         #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1193     dx = dx + (r2l and 0 or curr.width/factor*expand)
1194 elseif curr.replace then
1195     local width = node.dimensions(curr.replace)/factor
1196     dx = dx - (r2l and width or 0)
1197     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1198     dx = dx + (r2l and 0 or width)
1199 elseif curr.id == node.id"rule" then
1200     local wd, ht, dp = getrulemetric(box, curr, true)
1201     if wd ~= 0 then
1202         local hd = ht + dp
1203         dx = dx - (r2l and wd or 0)
1204         if hd ~= 0 and curr.subtype == 0 then
1205             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1206         end
1207         dx = dx + (r2l and 0 or wd)
1208     end
1209 elseif curr.id == node.id"glue" then
1210     local width = node.effective_glue(curr, box)/factor
1211     dx = dx - (r2l and width or 0)
1212     if curr.leader then
1213         local curr, kind = curr.leader, curr.subtype
1214         if curr.id == node.id"rule" then
1215             local wd, ht, dp = getrulemetric(box, curr, true)
1216             local hd = ht + dp
1217             if hd ~= 0 then
1218                 wd = width
1219                 if wd ~= 0 and curr.subtype == 0 then
1220                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1221                 end
1222             end
1223         elseif curr.head then

```

```

1224     local wd = curr.width/factor
1225     if wd <= width then
1226         local dx = r2l and dx+width or dx
1227         local n, ix = 0, 0
1228         if kind == 100 or kind == 103 then -- todo: gleaders
1229             local adx = abs(dx-dirs[1].dx)
1230             local ndx = math.ceil(adx / wd) * wd
1231             local diff = ndx - adx
1232             n = math.floor((width-diff) / wd)
1233             dx = dx + (r2l and -diff-wd or diff)
1234         else
1235             n = math.floor(width / wd)
1236             if kind == 101 then
1237                 local side = width % wd / 2
1238                 dx = dx + (r2l and -side-wd or side)
1239             elseif kind == 102 then
1240                 ix = width % wd / (n+1)
1241                 dx = dx + (r2l and -ix-wd or ix)
1242             end
1243         end
1244         wd = r2l and -wd or wd
1245         ix = r2l and -ix or ix
1246         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1247         for i=1,n do
1248             res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1249             dx = dx + wd + ix
1250         end
1251     end
1252 end
1253 end
1254 dx = dx + (r2l and 0 or width)
1255 elseif curr.id == node.id"kern" then
1256     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1257 elseif curr.id == node.id"math" then
1258     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1259 elseif curr.id == node.id"vlist" then
1260     dx = dx - (r2l and curr.width/factor or 0)
1261     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1262     dx = dx + (r2l and 0 or curr.width/factor)
1263 elseif curr.id == node.id"hlist" then
1264     dx = dx - (r2l and curr.width/factor or 0)
1265     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1266     dx = dx + (r2l and 0 or curr.width/factor)
1267 end
1268 curr = node.getnext(curr)
1269 end
1270 return res
1271 end
1272 function luamplib.outlinetext (text)

```

```

1273     local fmt = process_tex_text(text)
1274     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1275     local box = texgetbox(id)
1276     local res = outline_horz({ }, box, box.head, 0, 0)
1277     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1278     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1279 end
1280 end
1281

```

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

```

1282 function luamplib.getunicodegraphemes (s)
1283     local t = { }
1284     local graphemes = require'lua-uni-graphemes'
1285     for _, _, c in graphemes.graphemes(s) do
1286         table.insert(t, c)
1287     end
1288     return t
1289 end
1290 function luamplib.unicodesubstring (s,b,e,grph)
1291     local tt, t, step = { }
1292     if grph then
1293         t = luamplib.getunicodegraphemes(s)
1294     else
1295         t = { }
1296         for _, c in utf8.codes(s) do
1297             table.insert(t, utf8.char(c))
1298         end
1299     end
1300     if b <= e then
1301         b, step = b+1, 1
1302     else
1303         e, step = e+1, -1
1304     end
1305     for i = b, e, step do
1306         table.insert(tt, t[i])
1307     end
1308     s = table.concat(tt):gsub("'", "'&ditto'")
1309     return string.format("%s", s)
1310 end
1311

```

METAPOST preambles

```

1312 luamplib.preambles = {
1313     preamble = [[
1314 boolean mplib ; mplib := true ;
1315 let dump = endinput ;
1316 let normalfontsize = fontsize;
1317 input %s ;
1318 ]],

```

```

1319 mplibcode = [[
1320 texscriptmode := 2;
1321 def rawtexttext primary t = runscript("luamplibtext{"&t&"}") enddef;
1322 def mplibcolor primary t = runscript("luamplibcolor{"&t&"}") enddef;
1323 def mplibdimen primary t = runscript("luamplibdimen{"&t&"}") enddef;
1324 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&"}") enddef;
1325 if known context_mlib:
1326   defaultfont := "cmtt10";
1327   let infont = normalinfont;
1328   let fontsize = normalfontsize;
1329   vardef thelabel@#(expr p,z) =
1330     if string p :
1331       thelabel@#(p infont defaultfont scaled defaultscale,z)
1332     else :
1333       p shifted (z + labeloffset*mfun_laboff@# -
1334         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1335         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1336     fi
1337   enddef;
1338 else:
1339   vardef texttext@# primary t = rawtexttext (t) enddef;
1340   def message expr t =
1341     if string t: runscript("mp.report[="&t&"]="") else: errmessage "Not a string" fi
1342   enddef;
1343   def withtransparency (expr a, t) =
1344     withprescript "tr_alternative=" & if numeric a: decimal fi a
1345     withprescript "tr_transparency=" & decimal t
1346   enddef;
1347   vardef ddecimal primary p =
1348     decimal xpart p & " " & decimal ypart p
1349   enddef;
1350   vardef boundingbox primary p =
1351     if (path p) or (picture p) :
1352       llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1353     else :
1354       origin
1355     fi -- cycle
1356   enddef;
1357 fi
1358 def resolvedcolor(expr s) =
1359   runscript("return luamplib.shadecolor('"&s&"')")
1360 enddef;
1361 def colordecimals primary c =
1362   if cmykcolor c:
1363     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1364     decimal yellowpart c & ":" & decimal blackpart c
1365   elseif rgbcolor c:
1366     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1367   elseif string c:

```

```

1368   if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1369   else:
1370     decimal c
1371   fi
1372 enddef;
1373 def externalfigure primary filename =
1374   draw rawtexttext("\includegraphics{"& filename &}")
1375 enddef;
1376 def TEX = texttext enddef;
1377 def mplibtexcolor primary c =
1378   runscript("return luamplib.gettexcolor('"& c &"')")
1379 enddef;
1380 def mplibrgbtexcolor primary c =
1381   runscript("return luamplib.gettexcolor('"& c &"', 'rgb')")
1382 enddef;
1383 def mplibgraphicstext primary t =
1384   begingroup;
1385   mplibgraphicstext_ (t)
1386 enddef;
1387 def mplibgraphicstext_ (expr t) text rest =
1388   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1389   fb, fc, dc, graphicstextpic, alsoordoublepath;
1390   picture graphicstextpic; graphicstextpic := nullpicture;
1391   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1392   let scale = scaled;
1393   def fakebold primary c = hide(fb:=c;) enddef;
1394   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1395   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1396   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1397   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1398   addto graphicstextpic alsoordoublepath (origin--cycle) rest; graphicstextpic:=nullpicture;
1399   def fakebold primary c = enddef;
1400   let fillcolor = fakebold; let drawcolor = fakebold;
1401   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1402   image(draw runscript("return luamplib.graphicstext([===["&t&"]===], "
1403     & decimal fb & ", "& fc & ", "& dc & "')") rest;)
1404   endgroup;
1405 enddef;
1406 def mplibglyph expr c of f =
1407   runscript (
1408     "return luamplib.glyph('"
1409     & if numeric f: decimal fi f
1410     & " ', '"
1411     & if numeric c: decimal fi c
1412     & " ')"
1413   )
1414 enddef;
1415 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1416 def mplibdrawglyph expr g =

```

```

1417 luamplib_tmp_num_ := 0;
1418 for item within g:
1419   fill pathpart item
1420   if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1421 endfor
1422 enddef;
1423 let mplibfillglyph = mplibdrawglyph;
1424 def mplibstrokeglyph expr g =
1425   luamplib_tmp_num_ := 0;
1426   for item within g:
1427     draw pathpart item
1428     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1429   endfor
1430 enddef;
1431 def mplibfillandstrokeglyph expr g =
1432   luamplib_tmp_num_ := 0;
1433   for item within g:
1434     draw pathpart item withpostscript
1435     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1436   endfor
1437 enddef;
1438 def withmplibcolors (expr f, s) =
1439   runscript("return luamplib.fillandstrokecolor('" &
1440     if not string f: colordecimals fi f & "', '" &
1441     if not string s: colordecimals fi s & "'')")
1442 enddef;
1443 def withmplibopacities (expr a, f, s) =
1444   withprescript "tr_alternative=" & if numeric a: decimal fi a
1445   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1446 enddef;
1447 def mplib_do_outline_text_set_b (text f) (text d) text r =
1448   def mplib_do_outline_options_f = f enddef;
1449   def mplib_do_outline_options_d = d enddef;
1450   def mplib_do_outline_options_r = r enddef;
1451 enddef;
1452 def mplib_do_outline_text_set_f (text f) text r =
1453   def mplib_do_outline_options_f = f enddef;
1454   def mplib_do_outline_options_r = r enddef;
1455 enddef;
1456 def mplib_do_outline_text_set_u (text f) text r =
1457   def mplib_do_outline_options_f = f enddef;
1458 enddef;
1459 def mplib_do_outline_text_set_d (text d) text r =
1460   def mplib_do_outline_options_d = d enddef;
1461   def mplib_do_outline_options_r = r enddef;
1462 enddef;
1463 def mplib_do_outline_text_set_r (text d) (text f) text r =
1464   def mplib_do_outline_options_d = d enddef;
1465   def mplib_do_outline_options_f = f enddef;

```

```

1466 def mplib_do_outline_options_r = r enddef;
1467 enddef;
1468 def mplib_do_outline_text_set_n text r =
1469   def mplib_do_outline_options_r = r enddef;
1470 enddef;
1471 def mplib_do_outline_text_set_p = enddef;
1472 def mplib_fill_outline_text =
1473   for n=1 upto mpliboutlinenum:
1474     i:=0;
1475     for item within mpliboutlinepic[n]:
1476       i:=i+1;
1477       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1478       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1479     endfor
1480   endfor
1481 enddef;
1482 def mplib_draw_outline_text =
1483   for n=1 upto mpliboutlinenum:
1484     for item within mpliboutlinepic[n]:
1485       draw pathpart item mplib_do_outline_options_d;
1486     endfor
1487   endfor
1488 enddef;
1489 def mplib_filldraw_outline_text =
1490   for n=1 upto mpliboutlinenum:
1491     i:=0;
1492     for item within mpliboutlinepic[n]:
1493       i:=i+1;
1494       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1495         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1496       else:
1497         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1498       fi
1499     endfor
1500   endfor
1501 enddef;
1502 vardef mpliboutlinetext@# (expr t) text rest =
1503   save kind; string kind; kind := str @#;
1504   save i; numeric i;
1505   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1506   def mplib_do_outline_options_d = enddef;
1507   def mplib_do_outline_options_f = enddef;
1508   def mplib_do_outline_options_r = enddef;
1509   runscript("return luamplib.outlinetext[==["&t&"]==]");
1510   image ( addto currentpicture also image (
1511     if kind = "f":
1512       mplib_do_outline_text_set_f rest;
1513       mplib_fill_outline_text;
1514     elseif kind = "d":

```

```

1515     mplib_do_outline_text_set_d rest;
1516     mplib_draw_outline_text;
1517 elseif kind = "b":
1518     mplib_do_outline_text_set_b rest;
1519     mplib_fill_outline_text;
1520     mplib_draw_outline_text;
1521 elseif kind = "u":
1522     mplib_do_outline_text_set_u rest;
1523     mplib_filldraw_outline_text;
1524 elseif kind = "r":
1525     mplib_do_outline_text_set_r rest;
1526     mplib_draw_outline_text;
1527     mplib_fill_outline_text;
1528 elseif kind = "p":
1529     mplib_do_outline_text_set_p;
1530     mplib_draw_outline_text;
1531 else:
1532     mplib_do_outline_text_set_n rest;
1533     mplib_fill_outline_text;
1534 fi;
1535 ) mplib_do_outline_options_r; )
1536 enddef ;
1537 def withmppattern primary p =
1538   withprescript "mplibpattern=" & if numeric p: decimal fi p
1539 enddef;
1540 primarydef t withpattern p =
1541   image(
1542     if cycle t:
1543       fill
1544     else:
1545       draw
1546     fi
1547     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1548 enddef;
1549 vardef mplibtransformmatrix (text e) =
1550   save t; transform t;
1551   t = identity e;
1552   runscript("luamplib.transformmatrix = {"
1553     & decimal xpart t & ","
1554     & decimal ypart t & ","
1555     & decimal xpart t & ","
1556     & decimal ypart t & ","
1557     & decimal xpart t & ","
1558     & decimal ypart t & ","
1559     & "}");
1560 enddef;
1561 primarydef p withmaskinggroup s =
1562   if picture p:
1563     image(

```

```

1564     draw p;
1565     draw center p withprescript "mplibfadestate=stop";
1566   )
1567 else:
1568   p withprescript "mplibfadestate=stop"
1569 fi
1570 withprescript "mplibfadetype=masking"
1571 withprescript "mplibmaskname=" & s
1572 enddef;
1573 def withmaskingbgcolor expr c =
1574   withprescript "mplibmaskingbgcolor=" & colordecimals c
1575 enddef;
1576 primarydef p withfademethod s =
1577   if picture p:
1578     image(
1579       draw p;
1580       draw center p withprescript "mplibfadestate=stop";
1581     )
1582   else:
1583     p withprescript "mplibfadestate=stop"
1584   fi
1585   withprescript "mplibfadetype=" & s
1586   hide(mplib_shade_step := 1;)
1587   withprescript "sh_color_a=1"
1588   withprescript "sh_color_b=0"
1589   withprescript "mplibfadebbox=" &
1590     decimal (xpart llcorner p -1/4) & ":" &
1591     decimal (ypart llcorner p -1/4) & ":" &
1592     decimal (xpart urcorner p +1/4) & ":" &
1593     decimal (ypart urcorner p +1/4)
1594 enddef;
1595 def withfadevector (expr a,b) =
1596   withprescript "mplibfadevector=" &
1597     decimal xpart a & ":" &
1598     decimal ypart a & ":" &
1599     decimal xpart b & ":" &
1600     decimal ypart b
1601 enddef;
1602 let withfadecenter = withfadevector;
1603 def withfaderadius (expr a,b) =
1604   withprescript "mplibfaderadius=" &
1605     decimal a & ":" &
1606     decimal b
1607 enddef;
1608 def withfadebbox (expr a,b) =
1609   withprescript "mplibfadebbox=" &
1610     decimal xpart a & ":" &
1611     decimal ypart a & ":" &
1612     decimal xpart b & ":" &

```

```

1613     decimal ypart b
1614 enddef;
1615 primarydef p asgroup s =
1616   image(
1617     draw center p
1618     withprescript "mplibgroupbbox=" &
1619       decimal (xpart llcorner p -1/4) & ":" &
1620       decimal (ypart llcorner p -1/4) & ":" &
1621       decimal (xpart urcorner p +1/4) & ":" &
1622       decimal (ypart urcorner p +1/4)
1623     withprescript "gr_state=start"
1624     withprescript "gr_type=" & s;
1625     draw p withprescript "sh_in_xobj=yes";
1626     draw center p withprescript "gr_state=stop";
1627   )
1628 enddef;
1629 def withgroupbbox (expr a,b) =
1630   withprescript "mplibgroupbbox=" &
1631     decimal xpart a & ":" &
1632     decimal ypart a & ":" &
1633     decimal xpart b & ":" &
1634     decimal ypart b
1635 enddef;
1636 def withgroupname expr s =
1637   withprescript "mplibgroupname=" & s
1638 enddef;
1639 def usemplibgroup primary s =
1640   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s &"\endcsname}")
1641   shifted runscript("return luamplib.trgroupshifts['' & s & ''"]")
1642 enddef;
1643 path    mplib_shade_path ;
1644 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1645 numeric mplib_shade_fx, mplib_shade_fy ;
1646 numeric mplib_shade_lx, mplib_shade_ly ;
1647 numeric mplib_shade_nx, mplib_shade_ny ;
1648 numeric mplib_shade_dx, mplib_shade_dy ;
1649 numeric mplib_shade_tx, mplib_shade_ty ;
1650 primarydef p withshadingmethod m =
1651   p
1652   if picture p :
1653     withprescript "sh_operand_type=picture"
1654     if textual p or (length p > 1):
1655       withprescript "sh_transform=no"
1656       mplib_with_shade_method (boundingbox p, m)
1657     else:
1658       withprescript "sh_transform=yes"
1659       mplib_with_shade_method (pathpart p, m)
1660     fi
1661   else :

```

```

1662     withprescript "sh_transform=yes"
1663     mplib_with_shade_method (p, m)
1664 fi
1665 enddef;
1666 def mplib_with_shade_method (expr p, m) =
1667   hide(mplib_with_shade_method_analyze(p))
1668   withprescript "sh_domain=0 1"
1669   withprescript "sh_color=into"
1670   withprescript "sh_color_a=" & colordecimals white
1671   withprescript "sh_color_b=" & colordecimals black
1672   withprescript "sh_first=" & ddecimal point 0 of p
1673   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1674   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1675   if m = "linear" :
1676     withprescript "sh_type=linear"
1677     withprescript "sh_factor=1"
1678     withprescript "sh_center_a=" & ddecimal llcorner p
1679     withprescript "sh_center_b=" & ddecimal urcorner p
1680   elseif m = "coons":
1681     withprescript "sh_type=coons"
1682     withprescript "sh_transform=no"
1683   elseif m = "triangle":
1684     withprescript "sh_type=triangle"
1685     withprescript "sh_transform=no"
1686   elseif m = "lattice":
1687     withprescript "sh_type=lattice"
1688     withprescript "sh_transform=no"
1689   elseif m = "tensor":
1690     withprescript "sh_type=tensor"
1691     withprescript "sh_transform=no"
1692     withprescript "sh_tensor_path=" &
1693       ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1694       ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1695       ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1696       ddecimal 1/4[point 3 of p, point 1 of p]
1697   else :
1698     withprescript "sh_type=circular"
1699     withprescript "sh_factor=1.2"
1700     withprescript "sh_center_a=" & ddecimal center p
1701     withprescript "sh_center_b=" & ddecimal center p
1702     withprescript "sh_radius_a=" & decimal 0
1703     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1704   fi
1705 enddef;
1706 def withlatticeverticesperrow expr n =
1707   withprescript "sh_lattice_perrow=" & decimal n
1708 enddef;
1709 def withlatticeverticesdata (text t) =
1710   hide(luamplib_tmp_num_ := 0;)

```

```

1711 withprescript "sh_lattice_data=" &
1712 for item = t:
1713   if odd(incr luamplib_tmp_num_):
1714     if pair item: ddecimal fi item & " " &
1715     fi
1716   endfor ""
1717   hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1718   for item = t:
1719     if not odd(incr luamplib_tmp_num_):
1720       withshadingstep( withshadingcolors(item,item) )
1721     fi
1722   endfor
1723 enddef;
1724 def withtrianglepatchinit (expr p, a, b, c) =
1725   if string p: withprescript "sh_triangle_string=" & p fi
1726   hide(mplib_shade_step := 0;)
1727   if string p:
1728     withtrianglepatchnext(0, (0,0), a)
1729     withtrianglepatchnext(0, (0,0), b)
1730     withtrianglepatchnext(0, (0,0), c)
1731   else:
1732     withtrianglepatchnext(0, point 0 of p, a)
1733     withtrianglepatchnext(0, point 1 of p, b)
1734     withtrianglepatchnext(0, point 2 of p, c)
1735   fi
1736 enddef;
1737 def withtrianglepatchnext (expr f, p, a) =
1738   withshadingstep(
1739     withprescript "sh_triangle_edge=" & decimal mplib_shade_step & "=" & decimal f
1740     withprescript "sh_triangle_vertex=" & decimal mplib_shade_step & "=" &
1741     if string p: p else: ddecimal p fi
1742     withshadingcolors (a, a)
1743   )
1744 enddef;
1745 def withcoonspatchinit (expr p, a, b, c, d) =
1746   withprescript "sh_coons_path=" &
1747   if string p: p
1748   else:
1749     ddecimal point      0 of p & " " &
1750     ddecimal postcontrol 0 of p & " " &
1751     ddecimal precontrol  1 of p & " " &
1752     ddecimal point      1 of p & " " &
1753     ddecimal postcontrol 1 of p & " " &
1754     ddecimal precontrol  2 of p & " " &
1755     ddecimal point      2 of p & " " &
1756     ddecimal postcontrol 2 of p & " " &
1757     ddecimal precontrol  3 of p & " " &
1758     ddecimal point      3 of p & " " &
1759     ddecimal postcontrol 3 of p & " " &

```

```

1760     ddecimal precontrol 0 of p
1761   fi
1762   hide(mplib_shade_step := 0;)
1763   withshadingstep ( withshadingcolors (a, b) )
1764   withshadingstep ( withshadingcolors (b, c) )
1765   withshadingstep ( withshadingcolors (c, d) )
1766 enddef;
1767 def withtensorpatchinit (expr p, q, a, b, c, d) =
1768   withprescript "sh_coons_path=" &
1769     if string p: p & " " & q
1770     else:
1771       ddecimal point      0 of p & " " &
1772       ddecimal postcontrol 0 of p & " " &
1773       ddecimal precontrol  1 of p & " " &
1774       ddecimal point      1 of p & " " &
1775       ddecimal postcontrol 1 of p & " " &
1776       ddecimal precontrol  2 of p & " " &
1777       ddecimal point      2 of p & " " &
1778       ddecimal postcontrol 2 of p & " " &
1779       ddecimal precontrol  3 of p & " " &
1780       ddecimal point      3 of p & " " &
1781       ddecimal postcontrol 3 of p & " " &
1782       ddecimal precontrol  0 of p & " " &
1783       ddecimal point      0 of q & " " &
1784       ddecimal point      1 of q & " " &
1785       ddecimal point      2 of q & " " &
1786       ddecimal point      3 of q
1787     fi
1788   hide(mplib_shade_step := 0;)
1789   withshadingstep ( withshadingcolors (a, b) )
1790   withshadingstep ( withshadingcolors (b, c) )
1791   withshadingstep ( withshadingcolors (c, d) )
1792 enddef;
1793 def withcoonspatchnext (expr f, p, a, b) =
1794   withshadingstep(
1795     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1796     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1797       if string p: p
1798       else:
1799         ddecimal postcontrol 1 of p & " " &
1800         ddecimal precontrol  2 of p & " " &
1801         ddecimal point      2 of p & " " &
1802         ddecimal postcontrol 2 of p & " " &
1803         ddecimal precontrol  3 of p & " " &
1804         ddecimal point      3 of p & " " &
1805         ddecimal postcontrol 3 of p & " " &
1806         ddecimal precontrol  0 of p
1807       fi
1808     withshadingcolors (a, b)

```

```

1809 )
1810 enddef;
1811 def withtensorpatchnext (expr f, p, q, a, b) =
1812   withshadingstep(
1813     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1814     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1815       if string p: p & " " & q
1816       else:
1817         ddecimal postcontrol 1 of p & " " &
1818         ddecimal precontrol 2 of p & " " &
1819         ddecimal point 2 of p & " " &
1820         ddecimal postcontrol 2 of p & " " &
1821         ddecimal precontrol 3 of p & " " &
1822         ddecimal point 3 of p & " " &
1823         ddecimal postcontrol 3 of p & " " &
1824         ddecimal precontrol 0 of q & " " &
1825         ddecimal point 0 of q & " " &
1826         ddecimal point 1 of q & " " &
1827         ddecimal point 2 of q & " " &
1828         ddecimal point 3 of q
1829       fi
1830     withshadingcolors (a, b)
1831   )
1832 enddef;
1833 def mplib_with_shade_method_analyze(expr p) =
1834   mplib_shade_path := p ;
1835   mplib_shade_step := 1 ;
1836   mplib_shade_fx := xpart point 0 of p ;
1837   mplib_shade_fy := ypart point 0 of p ;
1838   mplib_shade_lx := mplib_shade_fx ;
1839   mplib_shade_ly := mplib_shade_fy ;
1840   mplib_shade_nx := 0 ;
1841   mplib_shade_ny := 0 ;
1842   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1843   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1844   for i=1 upto length(p) :
1845     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1846     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1847     if mplib_shade_tx > mplib_shade_dx :
1848       mplib_shade_nx := i + 1 ;
1849       mplib_shade_lx := xpart point i of p ;
1850       mplib_shade_dx := mplib_shade_tx ;
1851     fi ;
1852     if mplib_shade_ty > mplib_shade_dy :
1853       mplib_shade_ny := i + 1 ;
1854       mplib_shade_ly := ypart point i of p ;
1855       mplib_shade_dy := mplib_shade_ty ;
1856     fi ;
1857   endfor ;

```

```

1858 enddef;
1859 vardef mplib_max_radius(expr p) =
1860   max (
1861     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1862     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1863     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1864     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1865   )
1866 enddef;
1867 def withshadingstep (text t) =
1868   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1869   withprescript "sh_step=" & decimal mplib_shade_step
1870   t
1871 enddef;
1872 let withfadestep = withshadingstep;
1873 def withshadingradius expr a =
1874   withprescript "sh_radius_a=" & decimal (xpart a)
1875   withprescript "sh_radius_b=" & decimal (ypart a)
1876 enddef;
1877 def withshadingorigin expr a =
1878   withprescript "sh_center_a=" & ddecimal a
1879   withprescript "sh_center_b=" & ddecimal a
1880 enddef;
1881 def withshadingvector expr a =
1882   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1883   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1884 enddef;
1885 def withshadingdirection expr a =
1886   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1887   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1888 enddef;
1889 def withshadingtransform expr a =
1890   withprescript "sh_transform=" & a
1891 enddef;
1892 def withshadingcenter expr a =
1893   withprescript "sh_center_a=" & ddecimal (
1894     center mplib_shade_path shifted (
1895       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1896       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1897     )
1898   )
1899 enddef;
1900 def withshadingcenters (expr a, b) =
1901   withprescript "sh_center_a=" & ddecimal a
1902   withprescript "sh_center_b=" & ddecimal b
1903   withshadingtransform "no"
1904   withshadingfactor 1
1905 enddef;
1906 let withshadingpoints = withshadingcenters;

```

```

1907 def withshadingextend (expr a, b) =
1908   withprescript "sh_extend=" &
1909     if a: "true" else: "false" fi & " " &
1910     if b: "true" else: "false" fi
1911 enddef;
1912 let withfadeextend = withshadingextend;
1913 def withshadingdomain expr d =
1914   withprescript "sh_domain=" & ddecimal d
1915 enddef;
1916 def withshadingfactor expr f =
1917   withprescript "sh_factor=" & decimal f
1918 enddef;
1919 def withshadingfraction expr a =
1920   if mplib_shade_step > 0 :
1921     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1922   fi
1923 enddef;
1924 let withfadefraction = withshadingfraction;
1925 def withshadingcolors (expr a, b) =
1926   if mplib_shade_step > 0 :
1927     withprescript "sh_color=into"
1928     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1929     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1930   else :
1931     withprescript "sh_color=into"
1932     withprescript "sh_color_a=" & colordecimals a
1933     withprescript "sh_color_b=" & colordecimals b
1934   fi
1935 enddef;
1936 let withfadeopacity = withshadingcolors;
1937 def withshadingstroke expr a =
1938   withprescript "sh_stroking=" & a
1939 enddef;
1940 def withshadingmatrix expr s =
1941   withprescript "sh_matrix=" & s
1942 enddef;
1943 let withfadematrix = withshadingmatrix;
1944 def mpliblength primary t =
1945   runscript("return utf8.len[==[" & t & "]==]")
1946 enddef;
1947 def mplibsubstring expr p of t =
1948   runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1949     & decimal xpart p & ",",
1950     & decimal ypart p & ")")
1951 enddef;
1952 def mplibbuclength primary t =
1953   runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
1954 enddef;
1955 def mplibbusubstring expr p of t =

```

```

1956 runscript("return luamplib.unicodesubstring([==[" & t & "]==],"
1957   & decimal xpart p & ","
1958   & decimal ypart p & ",true)")
1959 enddef;
1960 ]],
1961 legacyverbatimtex = [[
1962 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1963 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1964 let VerbatimTeX = specialVerbatimTeX;
1965 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1966   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1967 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1968   "runscript(" &ditto&
1969   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1970   "luamplib.in_the_fig=false" &ditto& ");";
1971 ]],
1972 texttextlabel = [[
1973 let luampliboriginalinfont = infont;
1974 primarydef s infont f =
1975   if (s < char 32)
1976     or (s = char 35) % #
1977     or (s = char 36) % $
1978     or (s = char 37) % %
1979     or (s = char 38) % &
1980     or (s = char 92) % \
1981     or (s = char 94) % ^
1982     or (s = char 95) % _
1983     or (s = char 123) % {
1984     or (s = char 125) % }
1985     or (s = char 126) % ~
1986     or (s = char 127) :
1987     s luampliboriginalinfont f
1988   else :
1989     rawtexttext(s)
1990   fi
1991 enddef;
1992 def fontsize expr f =
1993   begingroup
1994     save size; numeric size;
1995     size := mplibdimen("1em");
1996     if size = 0: 10pt else: size fi
1997   endgroup
1998 enddef;
1999 ]],
2000 }
2001

```

process\_mplibcode

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

```

2002 luamplib.verbatiminput = false
2003 luamplib.everymplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2004 luamplib.everyendmplib = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2005 function luamplib.process_mplibcode (data, instancename)
2006   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2007   if luamplib.legacyverbatim then
2008     luamplib.figid, tex_code_pre_mplib = 1, {}
2009   end
2010   local everymplib = luamplib.everymplib[instancename]
2011   local everyendmplib = luamplib.everyendmplib[instancename]
2012   data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
2013   :gsub("\r", "\n")

```

These five lines are needed for mplibverbatim mode.

```

2014   if luamplib.verbatiminput then
2015     data = data:gsub("\mpcolor%s+(-%b{})", "mplibcolor(\"%1\")")
2016     :gsub("\mpdim%s+(%b{})", "mplibdimen(\"%1\")")
2017     :gsub("\mpdim%s+(\\%a+)", "mplibdimen(\"%1\")")
2018     :gsub(btex_etex, "btex %1 etex ")
2019     :gsub(verbatimtex_etex, "verbatimtex %1 etex;")
2020   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.

```

2021     local t = { } -- to store btex, verbatimtex, string
2022     data = data:gsub(btex_etex, function(str)
2023       t[#t+1] = str
2024       return format("btex \\unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
2025     end)
2026     :gsub(verbatimtex_etex, function(str)
2027       t[#t+1] = str
2028       return format("verbatimtex \\unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
2029     end)
2030     :gsub('"(.)"', function(str)
2031       t[#t+1] = str
2032       return format("\"\\unexpanded{!l!u!a!%s!m!p!l!}\"", #t)
2033     end)
2034     :gsub("\\\\%", "\\0PerCent\\0")
2035     :gsub("%%.-\\n", "\\n")
2036     :gsub("%zPerCent%z", "\\%")
2037     run_tex_code(format("\mplibtmptoks\\expandafter{\\expanded{%s}}", data))
2038     data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

2039     :gsub("##", "#")
2040     :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2041   end

```

```

2042 process(data, instancename)
2043 end
2044

```

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

```

2045 local figcontents = { post = { } }
2046 local function put2output(a,...)
2047   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2048 end
2049 local function pdf_startfigure(n,llx,lly,urx,ury)
2050   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2051 end
2052 local function pdf_stopfigure()
2053   put2output("\mplibstoptoPDF")
2054 end

```

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

```

2055 local function pdf_literalcode (...)
2056   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2057 end
2058 local start_pdf_code = pdfmode
2059 and function() pdf_literalcode"q" end
2060 or function() put2output"\special{pdf:bcontent}" end
2061 local stop_pdf_code = pdfmode
2062 and function() pdf_literalcode"Q" end
2063 or function() put2output"\special{pdf:econtent}" end
2064

```

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

```

2065 local function put_tex_boxes (object,prescript)
2066   local box = prescript.mplibtexboxid:explode":"
2067   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2068   if n and tw and th then
2069     local op = object.path
2070     local first, second, fourth = op[1], op[2], op[4]
2071     local tx, ty = first.x_coord, first.y_coord
2072     local sx, rx, ry, sy = 1, 0, 0, 1
2073     if tw ~= 0 then
2074       sx = (second.x_coord - tx)/tw
2075       rx = (second.y_coord - ty)/tw
2076       if sx == 0 then sx = 0.00001 end
2077     end
2078     if th ~= 0 then
2079       sy = (fourth.y_coord - ty)/th
2080       ry = (fourth.x_coord - tx)/th
2081       if sy == 0 then sy = 0.00001 end
2082     end
2083

```

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

```

2084     local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2085     put2output("\\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2086 end
2087 end
2088

```

## Colors

```

2089 local do_preobj_CR
2090 do
2091     local prev_override_color
2092     function do_preobj_CR(object,prescript)
2093         if object.postscript == "collect" then return end
2094         local override = prescript and prescript.mpliboverridecolor
2095         if override then
2096             if pdfmode then
2097                 pdf_literalcode(override)
2098                 override = nil
2099             else
2100                 put2output("\\special{%s}",override)
2101                 prev_override_color = override
2102             end
2103         else
2104             local cs = object.color
2105             if cs and #cs > 0 then
2106                 pdf_literalcode(luamplib.colorconverter(cs))
2107                 prev_override_color = nil
2108             elseif not pdfmode then
2109                 override = prev_override_color
2110                 if override then
2111                     put2output("\\special{%s}",override)
2112                 end
2113             end
2114         end
2115         return override
2116     end
2117 end
2118

```

## For transparency, shading, fading, and pattern

```

2119 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2120 local pdfobjs, pdfetcs = {}, {}
2121 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
2122 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2123 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
2124 local update_pdfobjs
2125 if pdfmode then
2126     function update_pdfobjs (os, stream)
2127         local key = os
2128         if stream then key = key..stream end
2129         local on = key and pdfobjs[key]

```

```

2130     if on then
2131         return on,false
2132     end
2133     if stream then
2134         on = pdf.immediateobj("stream",stream,os)
2135     elseif os then
2136         on = pdf.immediateobj(os)
2137     else
2138         on = pdf.reserveobj()
2139     end
2140     if key then
2141         pdfobjs[key] = on
2142     end
2143     return on,true
2144 end
2145 else
2146     function update_pdfobjs (os, stream)
2147         local key = os
2148         if stream then key = key..stream end
2149         local on = key and pdfobjs[key]
2150         if on then
2151             return on,false
2152         end
2153         on = pdfetcs.cnt or 1
2154         if stream then
2155             texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2156         elseif os then
2157             texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2158         else
2159             texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2160         end
2161         pdfetcs.cnt = on + 1
2162         if key then
2163             pdfobjs[key] = on
2164         end
2165         return on,true
2166     end
2167 end
2168 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2169 if pdfmode then
2170     pdfetcs.getpageres = pdf.getpageresources or function() return pdf.pageresources end
2171     local getpageres = pdfetcs.getpageres
2172     local setpageres = pdf.setpageresources or function(s) pdf.pageresources = s end
2173     local initialize_resources = function (name)
2174         local tabname = format("%s_res",name)
2175         pdfetcs[tabname] = { }
2176         if luatexbase.callbacktypes.finish_pdffile then -- ltuatex
2177             local obj = pdf.reserveobj()
2178             setpageres(format("%s/%s %i 0 R", getpageres() or "", name, obj))

```

```

2179     luatexbase.add_to_callback("finish_pdffile", function()
2180         pdf.immediateobj(obj, format("<< %s >>", tableconcat(pdfetcs[tabname])))
2181     end,
2182     format("luamplib.%s.finish_pdffile", name))
2183 end
2184 end
2185 pdfetcs.fallback_update_resources = function (name, res)
2186     local tabname = format("%s_res", name)
2187     if not pdfetcs[tabname] then
2188         initialize_resources(name)
2189     end
2190     if luatexbase.callbacktypes.finish_pdffile then
2191         local t = pdfetcs[tabname]
2192         t[#t+1] = res
2193     else
2194         local tpr, n = getpagemres() or "", 0
2195         tpr, n = tpr:gsub(format("/%s<<", name), "%1"..res)
2196         if n == 0 then
2197             tpr = format("%s/%s<< %s >>", tpr, name, res)
2198         end
2199         setpagemres(tpr)
2200     end
2201 end
2202 else
2203     texsprint {
2204         "\\luamplibatfirstshipout{",
2205         "\\special{pdf:obj @MPlibTr<<>>}",
2206         "\\special{pdf:obj @MPlibSh<<>>}",
2207         "\\special{pdf:obj @MPlibCS<<>>}",
2208         "\\special{pdf:obj @MPlibPt<<>>}}",
2209     }
2210 pdfetcs.fallback_update_resources = function (name, res, obj)
2211     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2212     local tabname = format("%s_res", name)
2213     if not pdfetcs[tabname] then
2214         texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2215         pdfetcs[tabname] = { }
2216     end
2217     tableinsert(pdfetcs[tabname], res)
2218 end
2219 end
2220

```

## Transparency

```

2221 local function add_extgs_resources (on, new)
2222     local key = format("MPlibTr%s", on)
2223     if new then
2224         local val = format(pdfetcs.resfmt, on)
2225         if pdfmanagement then

```

```

2226     texsprint {
2227         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{", val, "}"
2228     }
2229     else
2230         local tr = format("/%s %s", key, val)
2231         if is_defined(pdfetcs.pgfbextgs) then
2232             texsprint { "\\csname ", pdfetcs.pgfbextgs, "\\endcsname{", tr, "}" }
2233         elseif is_defined"TRP@list" then
2234             texsprint(catat11,{
2235                 [[\if@files\immediate\write\@auxout{]],
2236                 [[\string\g@addto@macro\string\TRP@list{]],
2237                 tr,
2238                 [[}]\fi]],
2239             })
2240             if not get_macro"TRP@list":find(tr) then
2241                 texsprint(catat11,[[\global\TRP@runtrue]])
2242             end
2243         else
2244             pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2245         end
2246     end
2247 end
2248 return key
2249 end
2250
2251 local do_preobj_TR
2252 do
2253     local transparency_modes = {
2254         [0] = "Normal",
2255         "Normal",      "Multiply",      "Screen",      "Overlay",
2256         "SoftLight",   "HardLight",     "ColorDodge",  "ColorBurn",
2257         "Darken",      "Lighten",      "Difference",   "Exclusion",
2258         "Hue",          "Saturation",   "Color",        "Luminosity",
2259         "Compatible",
2260         normal         = "Normal",      multiply = "Multiply",  screen   = "Screen",
2261         overlay        = "Overlay",      softlight = "SoftLight", hardlight = "HardLight",
2262         colordodge     = "ColorDodge",    colorburn = "ColorBurn", darken    = "Darken",
2263         lighten        = "Lighten",       difference = "Difference", exclusion = "Exclusion",
2264         hue             = "Hue",          saturation = "Saturation", color      = "Color",
2265         luminosity     = "Luminosity",    compatible = "Compatible",
2266     }
2267     function do_preobj_TR(object,prescript)
2268         if object.postscript == "collect" then return end
2269         local opaq = prescript and prescript.tr_transparency
2270         if not opaq then return end
2271
2272         local key, on, os, new
2273         local mode = prescript.tr_alternative or 1
2274         mode = transparency_modes[tonumber(mode) or mode:lower()]

```

```

2275   if not mode then
2276     mode = prescript.tr_alternative
2277     warn("unsupported blend mode: '%s'", mode)
2278   end
2279   opaq = opaq:explode":""
2280   for i,v in ipairs(opaq) do
2281     opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2282   end
2283   for i,v in ipairs[ {mode,opaq[1],opaq[2] or opaq[1]}, {"Normal",1,1} ] do
2284     os = format("</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2285     on, new = update_pdfobjs(os)
2286     key = add_extgs_resources(on,new)
2287     if i == 1 then
2288       pdf_literalcode("/%s gs",key)
2289     else
2290       return format("/%s gs",key)
2291     end
2292   end
2293 end
2294 end
2295

```

Shading with *metafun* format.

```

2296 local function add_shading_resources (on, new)
2297   if new then
2298     local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2299     if pdfmanagement then
2300       texsprint {
2301         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2302       }
2303     else
2304       local res = format("/%s %s", key, val)
2305       pdfetcs.fallback_update_resources("Shading",res,"@MPlibSh")
2306     end
2307   end
2308 end
2309 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2310   for _,v in ipairs{ca,cb} do
2311     for i,vv in ipairs(v) do
2312       for ii,vvv in ipairs(vv) do
2313         v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2314       end
2315     end
2316   end
2317   local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2318   if steps > 1 then
2319     local list,bounds,encode = { },{ },{ }
2320     for i=1,steps do
2321       if i < steps then

```

```

2322     bounds[i] = format("%.3f", fractions[i] or 1)
2323 end
2324 encode[2*i-1] = 0
2325 encode[2*i] = 1
2326 os = fun2fmt:format(domain,tableconcat(ca[i],' '),tableconcat(cb[i],' '))
2327 :gsub(decimals,rmzeros)
2328 list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2329 end
2330 os = tableconcat {
2331     "<</FunctionType 3",
2332     format("/Bounds[%s]", tableconcat(bounds,' ')),
2333     format("/Encode[%s]", tableconcat(encode,' ')),
2334     format("/Functions[%s]", tableconcat(list, ' ')),
2335     format("/Domain[%s]>>", domain),
2336 } :gsub(decimals,rmzeros)
2337 else
2338     os = fun2fmt:format(domain,tableconcat(ca[1],' '),tableconcat(cb[1],' '))
2339     :gsub(decimals,rmzeros)
2340 end
2341 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2342 os = tableconcat {
2343     format("<</ShadingType %i", shtype),
2344     format("/ColorSpace %s", colorspace),
2345     format("/Function %s", objref),
2346     format("/Coords[%s]", coordinates),
2347     format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2348 } :gsub(decimals,rmzeros)
2349 local on, new = update_pdfobjs(os)
2350 add_shading_resources(on, new)
2351 return on
2352 end
2353
2354 local function get_mp_matrix (matrix)
2355     process(("mplibtransformmatrix(%s);"):format(matrix), "@mplibtransformmatrix")
2356     return luamplib.transformmatrix
2357 end
2358
2359 local do_preobj_SH
2360 do
2361     pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2362         run_tex_code({
2363             [[\color_model_new:nnn]],
2364             format("{mplibcolorspace_%s}", names:gsub(",","_")),
2365             format("{DeviceN}{names={%s}}", names),
2366             [[\edef\mplib_atempa{\pdf_object_ref_last:}]],
2367         }, ccexplat)
2368         local colorspace = get_macro'mplib_atempa'
2369         t[names] = colorspace
2370         return colorspace

```

```

2371 end })
2372 local function color_normalize(ca,cb)
2373     if #cb == 1 then
2374         if #ca == 4 then
2375             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2376         else -- #ca = 3
2377             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2378         end
2379     elseif #cb == 3 then -- #ca == 4
2380         cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
2381     end
2382 end
2383 local function write_mesh_objs (shdtype, colorspace, stream, devicen, perrow)
2384     local cc = colorspace == "/DeviceCMYK" and 4
2385             or colorspace == "/DeviceRGB" and 3
2386             or devicen or 1
2387     local dict = format(
2388         "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2389         shdtype,
2390         perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2391         colorspace,
2392         (" 0 1"):rep(cc))
2393     local on, new
2394     if pdfmode then
2395         on, new = update_pdfobjs(dict, stream)
2396     else
2397         stream = format("%02X"):rep(stream:len()), stream:byte(1,stream:len())
2398         on, new = update_pdfobjs(format(
2399             "<< /Filter[/ASCIIHexDecode]/Length %d>>\nstream\n%s\nendstream",
2400             dict, stream:len(), stream))
2401     end
2402     add_shading_resources(on, new)
2403     return on
2404 end
2405 local function colors_ff (colortab, coons)
2406     local colors, devicen = { }
2407     for i,v in ipairs(colortab) do
2408         colors[i] = { }
2409         for ii,vv in ipairs(v) do
2410             if type(vv) == "string" and vv:find"%s" then
2411                 local t = vv:explode()
2412                 for iii,vvv in ipairs(t) do
2413                     colors[i][iii] = math.floor(vvv * 0xFF)
2414                 end
2415                 devicen = #t
2416             else
2417                 colors[i][ii] = math.floor(vv * 0xFF)
2418             end
2419         end
2420     end

```

```

2420     end
2421     if not coons then return colors, devicen end
2422     local t = { }
2423     for _,v in ipairs(colors) do
2424         for _,vv in ipairs(v) do
2425             t[#t+1] = vv
2426         end
2427     end
2428     return t, devicen
2429 end
2430 local function vertex_ffff (Xt, Yt)
2431     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2432     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2433     local wd, ht = xmax - xmin, ymax - ymin
2434     for i = 1, #Xt do
2435         Xt[i] = math.floor((Xt[i] - xmin)/wd * 0xFFFF )
2436     end
2437     for i = 1, #Yt do
2438         Yt[i] = math.floor((Yt[i] - ymin)/ht * 0xFFFF )
2439     end
2440     return Xt, Yt, xmin, ymin, wd, ht
2441 end
2442 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2443     local perrow = prescript.sh_lattice_perrow or 2
2444     local data = prescript.sh_lattice_data
2445     data = data and data:explode() or { }
2446     if #data < 8 then
2447         data = { }
2448         local path = object.path
2449         for _,i in ipairs{1,2,4,3} do
2450             data[#data+1] = path[i].x_coord
2451             data[#data+1] = path[i].y_coord
2452         end
2453     end
2454     if #data / 2 % perrow ~= 0 then err"vertices_per_row mismatches lattice_data" end
2455     local Xt, Yt = { }, { }
2456     for i = 1, #data, 2 do
2457         Xt[#Xt+1] = tonumber(data[i])
2458         Yt[#Yt+1] = tonumber(data[i+1])
2459     end
2460     local xmin, ymin, wd, ht
2461     Xt, Yt, xmin, ymin, wd, ht = vertex_ffff(Xt, Yt)
2462
2463     local colors, devicen = { }
2464     if #ca > 3 then
2465         colors, devicen = colors_ff(ca)
2466     elseif colorspace == "/DeviceRGB" then
2467         colors = {{255,0,0},{0,255,0},{0,0,255},{255,255,0}}
2468     end

```

```

2469
2470     local stream = { }
2471     for i = 1, #Xt do
2472         stream[#stream+1] = string.pack( ">HH", Xt[i], Yt[i])
2473         if not colors[i] then err"colorspace mismatch?" end
2474         for _,vv in ipairs(colors[i]) do
2475             stream[#stream+1] = string.char(tonumber(vv))
2476         end
2477     end
2478
2479     local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2480     local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2481     return on, matrix
2482 end
2483 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2484     local str = prescript.sh_triangle_string
2485     if str then
2486         str = str:explode()
2487         prescript.sh_triangle_vertex_1 = format("%s %s", str[1], str[2])
2488         prescript.sh_triangle_vertex_2 = format("%s %s", str[3], str[4])
2489         prescript.sh_triangle_vertex_3 = format("%s %s", str[5], str[6])
2490     end
2491
2492     local Xt, Yt = { }, { }
2493     local steps = tonumber(prescript.sh_step) or 0
2494     if steps > 2 then
2495         for i = 1, steps do
2496             local v = prescript["sh_triangle_vertex_" .. i]:explode()
2497             Xt[#Xt+1] = tonumber(v[1])
2498             Yt[#Yt+1] = tonumber(v[2])
2499         end
2500     else
2501         local path = object.path
2502         for i = 1, 3 do
2503             Xt[#Xt+1] = path[i].x_coord
2504             Yt[#Yt+1] = path[i].y_coord
2505         end
2506     end
2507     local xmin, ymin, wd, ht
2508     Xt, Yt, xmin, ymin, wd, ht = vertex_ffff(Xt, Yt)
2509
2510     local colors, devicen = { }
2511     if #ca > 2 then
2512         colors, devicen = colors_ff(ca)
2513     elseif colorspace == "/DeviceRGB" then
2514         colors = { {255, 0, 0}, {0, 255, 0}, {0, 0, 255} }
2515     end
2516
2517     local stream = { }

```

```

2518   for i = 1, #Xt do
2519     local flag = tonumber(prescript["sh_triangle_edge_" .. i]) or 0
2520     stream[#stream+1] = string.char(flag)
2521     stream[#stream+1] = string.pack( ">HH", Xt[i], Yt[i])
2522     if not colors[i] then err"colorspace mismatch?" end
2523     for _,vv in ipairs(colors[i]) do
2524       stream[#stream+1] = string.char(tonumber(vv))
2525     end
2526   end
2527
2528   local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2529   local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2530   return on, matrix
2531 end
2532 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2533   local tensor = prescript.sh_type == "tensor"
2534   local X_t, Y_t = { }, { }
2535   local path = prescript.sh_coons_path
2536   if path then
2537     for i,v in ipairs( path:explode() ) do
2538       local t = i % 2 == 1 and X_t or Y_t
2539       t[#t+1] = tonumber(v)
2540     end
2541   else
2542     path = object.path
2543     for i = 1, 4 do
2544       X_t[#X_t+1] = path[i].x_coord
2545       Y_t[#Y_t+1] = path[i].y_coord
2546       X_t[#X_t+1] = path[i].right_x
2547       Y_t[#Y_t+1] = path[i].right_y
2548       local j = i == #path and 1 or i+1
2549       X_t[#X_t+1] = path[j].left_x
2550       Y_t[#Y_t+1] = path[j].left_y
2551     end
2552   if tensor then
2553     for i,v in ipairs( prescript.sh_tensor_path:explode() ) do
2554       local t = i % 2 == 1 and X_t or Y_t
2555       t[#t+1] = tonumber(v)
2556     end
2557   end
2558 end
2559
2560 local steps = tonumber(prescript.sh_step) or 0
2561 for i = 4, steps do
2562   local path = prescript["sh_coons_path_"..i]
2563   for i,v in ipairs( path:explode() ) do
2564     local t = i % 2 == 1 and X_t or Y_t
2565     t[#t+1] = tonumber(v)
2566   end

```

```

2567     end
2568
2569     local xmin, ymin, wd, ht
2570     X_t, Y_t, xmin, ymin, wd, ht = vertex_ffff(X_t, Y_t)
2571
2572     local n = tensor and 16 or 12
2573     local coords = { }
2574     for i = 1, n do
2575         coords[#coords+1] = X_t[i]
2576         coords[#coords+1] = Y_t[i]
2577     end
2578     coords = string.pack( ">".("H"):rep(2*n), tableunpack(coords))
2579
2580     local colors, devicen
2581     if ca[3] and cb[3] then
2582         colors, devicen = colors_ff({ca[1], ca[2], ca[3], cb[3]}, true)
2583     elseif colorspace == "/DeviceRGB" then
2584         colors = { 255, 0, 0, 0, 255, 0, 0, 0, 255, 255, 255, 0 }
2585     end
2586     if not colors then err"colorspace mismatch?" end
2587     colors = string.char(tableunpack(colors))
2588
2589     local stream = { string.char(0) .. coords .. colors }
2590
2591     local nn = tensor and 12 or 8
2592     for i = 4, steps do
2593         local coords = { }
2594         for j = n+(i-4)*nn+1, n+(i-3)*nn do
2595             coords[#coords+1] = X_t[j]
2596             coords[#coords+1] = Y_t[j]
2597         end
2598         coords = string.pack( ">".("H"):rep(2*nn), tableunpack(coords))
2599
2600         if not ca[i] or not cb[i] then err"colorspace mismatch?" end
2601         local colors = colors_ff({ca[i], cb[i]}, true)
2602         colors = string.char(tableunpack(colors))
2603
2604         local flag = tonumber(prescript["sh_coons_edge_"..i])
2605
2606         stream[#stream+1] = string.char(flag)..coords..colors
2607     end
2608
2609     local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2610     local on = write_mesh_objs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2611     return on, matrix
2612 end
2613 function do_preobj_SH(object, prescript)
2614     local shade_no
2615     local sh_type = prescript and prescript.sh_type

```

```

2616     if not sh_type then return end
2617
2618     local domain = prescript.sh_domain or "0 1"
2619     local centera = (prescript.sh_center_a or "0 0"):explode()
2620     local centerb = (prescript.sh_center_b or "0 0"):explode()
2621     local transform = prescript.sh_transform == "yes"
2622     local sx,sy,sr,dx,dy = 1,1,1,0,0
2623     if transform then
2624         local first = (prescript.sh_first or "0 0"):explode()
2625         local setx = (prescript.sh_set_x or "0 0"):explode()
2626         local sety = (prescript.sh_set_y or "0 0"):explode()
2627         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2628         if x ~= 0 and y ~= 0 then
2629             local path = object.path
2630             local path1x = path[1].x_coord
2631             local path1y = path[1].y_coord
2632             local path2x = path[x].x_coord
2633             local path2y = path[y].y_coord
2634             local dxa = path2x - path1x
2635             local dya = path2y - path1y
2636             local dxb = setx[2] - first[1]
2637             local dyb = sety[2] - first[2]
2638             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2639                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
2640                 sy = dya / dyb ; if sy < 0 then sy = - sy end
2641                 sr = math.sqrt(sx^2 + sy^2)
2642                 dx = path1x - sx*first[1]
2643                 dy = path1y - sy*first[2]
2644             end
2645         end
2646     end
2647     local ca, cb, colorspace, steps, fractions
2648     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:" }
2649     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:" }
2650     steps = tonumber(prescript.sh_step) or 1
2651     if steps > 1 then
2652         fractions = { prescript.sh_fraction_1 or 0 }
2653         for i=2,steps do
2654             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2655             ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
2656             cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
2657         end
2658     end
2659     if prescript.mplib_spotcolor then
2660         ca, cb = { }, { }
2661         local names, pos, objref = { }, -1, ""
2662         local script = object.prescript:explode"\13+"
2663         for i=#script,1,-1 do
2664             if script[i]:find"mplib_spotcolor" then

```

```

2665     local t, name, value = script[i]:explode="[2]:explode":
2666     value, objref, name = t[1], t[2], t[3]
2667     if not names[name] then
2668         pos = pos+1
2669         names[name] = pos
2670         names[#names+1] = name
2671     end
2672     t = { }
2673     for j=1,names[name] do t[#t+1] = 0 end
2674     t[#t+1] = value
2675     tableinsert(#ca == #cb and ca or cb, t)
2676     end
2677 end
2678 for _,t in ipairs{ca,cb} do
2679     for _,tt in ipairs(t) do
2680         for i=1,#names-#tt do tt[#tt+1] = 0 end
2681     end
2682 end
2683 if #names == 1 then
2684     colorspace = objref
2685 else
2686     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2687 end
2688 else
2689     local model = 0
2690     for _,t in ipairs{ca,cb} do
2691         for _,tt in ipairs(t) do
2692             model = model > #tt and model or #tt
2693         end
2694     end
2695     for _,t in ipairs{ca,cb} do
2696         for _,tt in ipairs(t) do
2697             if #tt < model then
2698                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2699             end
2700         end
2701     end
2702     colorspace = model == 4 and "/DeviceCMYK"
2703                 or model == 3 and "/DeviceRGB"
2704                 or model == 1 and "/DeviceGray"
2705                 or err"unknown color model"
2706 end
2707 local extend = prescript.sh_extend
2708 local mesh_matrix
2709 if sh_type == "linear" then
2710     local coordinates = format("%f %f %f %f",
2711         dx + sx*centera[1], dy + sy*centera[2],
2712         dx + sx*centerb[1], dy + sy*centerb[2])
2713     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)

```

```

2714 elseif sh_type == "circular" then
2715     local factor = prescript.sh_factor or 1
2716     local radiusa = factor * prescript.sh_radius_a
2717     local radiusb = factor * prescript.sh_radius_b
2718     local coordinates = format("%f %f %f %f %f %f",
2719         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2720         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2721     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2722 elseif sh_type == "coons" or sh_type == "tensor" then
2723     shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2724 elseif sh_type == "triangle" then
2725     shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2726 elseif sh_type == "lattice" then
2727     shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2728 else
2729     err"unknown shading type"
2730 end
2731
2732 local matrix = prescript.sh_matrix
2733 if matrix and matrix:find"%a" then
2734     local t = get_mp_matrix(matrix)
2735     matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2736 end
2737 if mesh_matrix and matrix then
2738     local a, b = mesh_matrix:explode(), matrix:explode()
2739     matrix = format("%f %f %f %f %f %f",
2740         a[1]*b[1]+a[2]*b[3], a[1]*b[2]+a[2]*b[4], a[3]*b[1]+a[4]*b[3], a[3]*b[2]+a[4]*b[4],
2741         a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)
2742 end
2743
2744 return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix
2745 end
2746 end
2747

```

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

```

2748 local function add_pattern_resources (key, val)
2749     if pdfmanagement then
2750         texsprint {
2751             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"
2752         }
2753     else
2754         local res = format("/%s %s", key, val)
2755         if is_defined(pdfetcs.pgfpattern) then
2756             texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2757         else
2758             pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2759         end
2760     end

```

```

2761 end
2762 if pdfmode then
2763   function luamplib.dolatelua (on, os, matrix)
2764     local h, v = pdf.getpos()
2765     local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2766     matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2767     :gsub(decimals,rmzeros)
2768     pdf.obj(on, format("<<Matrix[%s]>>", os, matrix))
2769     pdf.refobj(on)
2770   end
2771 else
2772   pdfetcs.shadingpatterns = { }
2773   pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2774   function luamplib.dolatelua (on, kind, xobj)
2775     local h, v
2776     local t = pdfetcs.shadingpatterns[on] or { }
2777     local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2778       or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2779     if shift then
2780       h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2781     else
2782       h, v = pdf.getpos()
2783       h = format("%f", h/factor) :gsub(decimals,rmzeros)
2784       v = format("%f", v/factor) :gsub(decimals,rmzeros)
2785     end
2786     if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2787       warn"Rerun to get correct shading pattern"
2788     end
2789     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2790     local init = pdfetcs.shadingpatterninit_w
2791     if init then pdfetcs.shadingpatterninit_w = nil end
2792     local f = ioopen(name, init and "w" or "a")
2793     if f then
2794       f:write(("<<Matrix[%s]>>"):format(on, h, v))
2795       f:close()
2796     else
2797       err"cannot write a file. check the cache dir path"
2798     end
2799   end
2800 end
2801 local function do_preobj_shading (object, prescript)
2802   if not prescript or not prescript.sh_operand_type then return end
2803   local on,_,matrix = do_preobj_SH(object, prescript)
2804   local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2805   matrix = matrix or "1 0 0 1 0 0"
2806   if prescript.sh_in_xobj == "yes" then
2807     on = update_pdfobjs(("<<Matrix[%s]>>"):format(os, matrix))
2808     goto skip_latelua
2809   end

```

```

2810 on = update_pdfobjs()
2811 if pdfmode then
2812   put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"],[",matrix,"]]}}")
2813 else
2814   local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2815             or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2816   local init = pdfetcs.shadingpatterninit_r
2817   if init then
2818     pdfetcs.shadingpatterninit_r = nil
2819     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2820     local f = ioopen(name)
2821     if f then
2822       for line in f:lines() do
2823         local t = line:explode()
2824         pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2825       end
2826       f:close()
2827     end
2828   end

```

This seems to be needed for proper functioning:

```

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

2829   local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2830   local mt = matrix:explode()
2831   matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2832   texsprint{ "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2833             format(" <<%s/Matrix[%s]>>", os, matrix) }
2834   put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2835             xobj and ("'%s',[[%s]]"):format(xobj[1], xobj[2]))
2836 end
2837 ::skip_latelua::
2838 local key, val = format("MPlibPt%s", on), format(pdfetcs.resfmt, on)
2839 add_pattern_resources(key,val)
2840 pdf_literalcode("/Pattern cs/%s scn", key)

```

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

```

2841 prescript.sh_type = nil
2842 end
2843

```

### Tiling Patterns

```

2844 pdfetcs.patterns = { }
2845 local function gather_resources (optres, ispattern)
2846   local t = { }
2847   if pdfmanagement then
2848     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2849       local mytoks

```

```

2850 run_tex_code ({
2851   "\\mplibtmp toks\\expanded{",
2852   "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/",v,"}",
2853   "\\pdfdict_use:n{g__pdf_Core/Page/Resources/",v,"}", "}",
2854 },ccexplat)
2855 mytoks = texgettoks"mplibtmp toks"
2856 if not pdfmode then
2857   mytoks = mytoks:gsup("\\str_convert_pdfname:n{s*{(.-)}", "%1") -- why not expanded?
2858 end
2859 mytoks = mytoks and mytoks:gsup("^s*{(.-)}s*$", "%1")
2860 if mytoks and mytoks ~= "" then
2861   t[#t+1] = ("%s<<s>>"):format(v, mytoks)
2862 end
2863 end
2864 elseif is_defined(pdfetcs.pgftextgs) then
2865   run_tex_code "\\relax" -- flush tex.sprint queue
2866   if pdfmode then
2867     for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2868                       ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2869                       Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2870       local res = (get_macro(v) or ""):gsup("^s*{(.-)}s*$", "%1")
2871       if res ~= "" then
2872         t[#t+1] = ("%s<<s>>"):format(k, res )
2873       end
2874     end
2875   else
2876     local abc = get_macro"pgfutil@abc" or ""
2877     for k,v in pairs { ExtGState = "@pgftextgs",
2878                       ColorSpace = "@pgfcolorspaces",
2879                       Pattern = "@pgfpatterns", } do
2880       local tt = { }
2881       for vv in abc:gmatch( v .. "s*{(%b<>)}" ) do
2882         tt[#tt+1] = vv:match("^<<s*{(.-)}s*>>$")
2883       end
2884       if #tt > 0 then
2885         t[#t+1] = ("%s<<s>>"):format(k, tableconcat(tt) )
2886       end
2887     end
2888   end

```

We still have to deal with Shading resources.

```

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

```

```

2897     t[#t+1] = res
2898   end
2899 end
2900 else
2901   if ispattern and is_defined"TRP@list" then

```

We do not gather transparent package's \TRP@list as Acrobat glitches on tiling pattern plus masking group, so warn users and recommend \DocumentMetadata

```

2902     warn"transparent package is not fully functional without pdfmanagement code."
2903   end
2904   if luatexbase.callbacktypes.finish_pdffile then
2905     for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2906       local tt = pdfetcs[v.."_res"]
2907       if tt then
2908         t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2909       end
2910     end
2911   else
2912     local res = pdfetcs.getpageres()
2913     if res then
2914       t[#t+1] = res
2915     end
2916   end
2917 end
2918 local result = tableconcat(t)
2919 if optres ~= "" then
2920   for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2921     local res = optres:match("/"..v.."*%b<>")
2922     if res then
2923       if result:find("/"..v) then
2924         res = res:match("<<(.+)>>$")
2925         result = result:gsub("/"..v.."*%b<<","%1"..res, 1)
2926       else
2927         result = result .. res
2928       end
2929     end
2930   end
2931 end
2932 return result
2933 end
2934 function luamplib.registerpattern ( boxid, name, opts )
2935   local box = texgetbox(boxid)
2936   local wd = format("%.3f",box.width/factor)
2937   local hd = format("%.3f",(box.height+box.depth)/factor)
2938   info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2939   if opts.xstep == 0 then opts.xstep = nil end
2940   if opts.ystep == 0 then opts.ystep = nil end
2941   if opts.colored == nil then
2942     opts.colored = opts.coloured

```

```

2943   if opts.colored == nil then
2944     opts.colored = true
2945   end
2946 end
2947 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2948 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2949 if opts.matrix and opts.matrix:find"%a" then
2950   local t = get_mp_matrix(opts.matrix)
2951   opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2952   opts.xshift = opts.xshift or format("%f", t[5])
2953   opts.yshift = opts.yshift or format("%f", t[6])
2954 end
2955 local attr = {
2956   "/Type/Pattern",
2957   "/PatternType 1",
2958   format("/PaintType %i", opts.colored and 1 or 2),
2959   "/TilingType 2",
2960   format("/XStep %s", opts.xstep or wd),
2961   format("/YStep %s", opts.ystep or hd),
2962   format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2963 }
2964 local optres = opts.resources or ""
2965 optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2966 local patterns = pdfetcs.patterns
2967 if pdfmode then
2968   if opts.bbox then
2969     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2970   end
2971   attr = tableconcat(attr) :gsub(decimals, rmzeros)
2972   local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2973   patterns[name] = { id = index, colored = opts.colored }
2974 else
2975   local cnt = #patterns + 1
2976   local objname = "@mplibpattern" .. cnt
2977   local metric = format("bbox %s", opts.bbox or format("0 0 %s %s", wd, hd))
2978   texsprint {
2979     "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2980     "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2981     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2982     "\\special{pdf:bcontent}",
2983     "\\special{pdf:bxobj ", objname, " ", metric, "}",
2984     "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2985     "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2986     "\\special{pdf:put @resources <<", optres, ">>}",
2987     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2988     "\\special{pdf:econtent}}",
2989   }
2990   patterns[cnt] = objname
2991   patterns[name] = { id = cnt, colored = opts.colored }

```

```

2992     patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
2993 end
2994 end
2995
2996 local do_preobj_PAT
2997 do
2998   local function pattern_colorspace (cs)
2999     local on, new = update_pdfobjs(format("/Pattern %s]", cs))
3000     if new then
3001       local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
3002       if pdfmanagement then
3003         texsprint {
3004           "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{" , val, "}"
3005         }
3006       else
3007         local res = format("/%s %s", key, val)
3008         if is_defined(pdfetcs.pgfcOLORSPACE) then
3009           texsprint { "\\csname ", pdfetcs.pgfcOLORSPACE, "\\endcsname{" , res, "}" }
3010         else
3011           pdfetcs.fallback_update_resources("ColorSpace",res,"@MPlibCS")
3012         end
3013       end
3014     end
3015     return on
3016   end
3017   function do_preobj_PAT(object, prescript)
3018     local name = prescript and prescript.mplibpattern
3019     if not name then return end
3020     local patterns = pdfetcs.patterns
3021     local patt = patterns[name]
3022     local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3023     local key = format("MPlibPt%s",index)
3024     if patt.colored then
3025       pdf_literalcode("/Pattern cs /%s scn", key)
3026     else
3027       local color = prescript.mpliboverridecolor
3028       if not color then
3029         local t = object.color
3030         color = t and #t>0 and luamplib.colorconverter(t)
3031       end
3032       if not color then return end
3033       local cs
3034       if color:find" cs " or color:find"@pdf.obj" then
3035         local t = color:explode()
3036         if pdfmode then
3037           cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
3038           color = t[3]
3039         else
3040           cs = t[2]

```

```

3041     color = t[3]:match"%[(.+)%]"
3042     end
3043     else
3044         local t = colorsplit(color)
3045         cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
3046         color = tableconcat(t, " ")
3047     end
3048     pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3049 end
3050 if not patt.done then
3051     local val = pdfmode and format("%s 0 R", index) or patterns[index]
3052     add_pattern_resources(key, val)
3053 end
3054 patt.done = true
3055 end
3056 end
3057

```

### Fading

```

3058 pdfetcs.fading = { }
3059 local function do_preobj_FADE (object, prescript)
3060     local fd_type = prescript and prescript.mplibfadetype
3061     local fd_stop = prescript and prescript.mplibfadestate
3062     if not fd_type then
3063         return fd_stop -- returns "stop" (if picture) or nil
3064     end
3065     local on, os, new
3066     if fd_type == "masking" then
3067         local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3068         on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.+)}")
3069         local bc = prescript.mplibmaskingbgcolor
3070         bc = bc and bc:gsub(":", " ")
3071         bc = bc and ("BC[%s]"):format(bc):gsub(decimals, rmzeros) or ""
3072         os = format("<</SMask<</S/Luminosity/G %s>>>>",
3073             pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3074     else
3075         local bbox = prescript.mplibfadebbox:explode":"
3076         local dx, dy = -bbox[1], -bbox[2]
3077         local vec = prescript.mplibfadevector; vec = vec and vec:explode":"
3078         if not vec then
3079             if fd_type == "linear" then
3080                 vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3081             else
3082                 local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3083                 vec = {centerx, centery, centerx, centery} -- center for both circles
3084             end
3085         end
3086         local coords = { vec[1], vec[2], vec[3], vec[4] }
3087         if fd_type == "linear" then

```

```

3088     coords = format("%f %f %f %f", tableunpack(coords))
3089 elseif fd_type == "circular" then
3090     local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3091     local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":"
3092     tableinsert(coords, 3, radius[1])
3093     tableinsert(coords, radius[2])
3094     coords = format("%f %f %f %f %f %f", tableunpack(coords))
3095 else
3096     err("unknown fading method '%s'", fd_type)
3097 end
3098 fd_type = fd_type == "linear" and 2 or 3
3099 local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3100 local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":" }
3101 local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":" }
3102 if steps > 1 then
3103     fractions = { prescript.sh_fraction_1 or 0 }
3104     for i=2,steps do
3105         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3106         ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":"
3107         cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":"
3108     end
3109 end
3110 local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3111 matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3112 matrix[5] = matrix[5] + dx
3113 matrix[6] = matrix[6] + dy
3114 matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3115 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3116 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3117 on = update_pdfobjs(os)
3118 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3119 local streamtext = format("q /Pattern cs/MPlibFd%s scn %s re f Q", on, bbox)
3120 :gsub(decimals,rmzeros)
3121 os = format("<</Pattern<</MPlibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3122 on = update_pdfobjs(os)
3123 local resources = format(pdfetcs.resfmt, on)
3124 on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
3125 local attr = tableconcat{
3126     "/Subtype/Form",
3127     "/BBox[" .. bbox .. "]",
3128     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
3129     "/Resources " .. resources,
3130     "/Group " .. format(pdfetcs.resfmt, on),
3131 } :gsub(decimals,rmzeros)
3132 on = update_pdfobjs(attr, streamtext)
3133 os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
3134 end
3135 on, new = update_pdfobjs(os)
3136 local key = add_extgs_resources(on,new)

```

```

3137 start_pdf_code()
3138 pdf_literalcode("/%s gs", key)
3139 if fd_stop then return "standalone" end
3140 return "start"
3141 end
3142

```

## Transparency Group

```

3143 pdfetcs.tr_group = { shifts = { } }
3144 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3145 local function do_preobj_GRP (object, prescript)
3146   local grstate = prescript and prescript.gr_state
3147   if not grstate then return end
3148   local trgroup = pdfetcs.tr_group
3149   if grstate == "start" then
3150     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3151     trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3152     trgroup.wrapped = false
3153     for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+)") do
3154       trgroup[v] = true
3155     end
3156     trgroup.bbox = prescript.mplibgroupbbox:explode":"
3157     put2output[["\begingroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3158   elseif grstate == "stop" then
3159     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3160     put2output(tableconcat{
3161       "\\egroup",
3162       format("\\wd\mplibscratchbox %fbp", urx-llx),
3163       format("\\ht\mplibscratchbox %fbp", ury-lly),
3164       "\\dp\mplibscratchbox 0pt",
3165     })
3166     local grattr
3167     if trgroup.off then
3168       grattr = ""
3169     else
3170       local on = update_pdfobjs(format("<</S/Transparency/I %s/K %s>>",
3171                                     trgroup.isolated, trgroup.knockout))
3172       grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3173     end
3174     local res = gather_resources("")
3175     local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3176     if pdfmode then
3177       if trgroup.wrapped then
3178         put2output(tableconcat{
3179           "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3180           "/BBox[" .. bbox .. "]} resources{" .. res .. "}" .. "\\mplibscratchbox",
3181           "\\setbox\\mplibscratchbox\\hbox{" .. "\\useboxresource\\lastsavedboxresourceindex}",
3182         })
3183       end

```

```

3184 put2output(tableconcat{
3185     "\\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3186     "/BBox[" , bbox , "]" , grattr , "} resources{" , res , "}\\mplibscratchbox",
3187     "\\luamplibtagasgroupput{" , trgroup.name , "}" ,
3188     [[\\setbox\\mplibscratchbox\\hbox{\\useboxresource\\lastsavedboxresourceindex}]],
3189     [[\\wd\\mplibscratchbox 0pt\\ht\\mplibscratchbox 0pt\\dp\\mplibscratchbox 0pt]],
3190     [[\\box\\mplibscratchbox]],
3191     "}\\endgroup",
3192     "\\expandafter\\xdef\\csname luamplib.group." , trgroup.name , "\\endcsname{" ,
3193     "\\setbox\\mplibscratchbox\\hbox{\\hskip" , -llx , "bp\\raise" , -lly , "bp\\hbox{" ,
3194     "\\useboxresource \\the\\lastsavedboxresourceindex" ,
3195     "}}\\wd\\mplibscratchbox" , urx-llx , "bp\\ht\\mplibscratchbox" , ury-lly , "bp" ,
3196     "\\box\\mplibscratchbox}" ,
3197 })
3198 else
3199     if trgroup.wrapped then
3200         trgroup.cnt = (trgroup.cnt or 0) + 1
3201         local objname = format("@mplibtrgr%s" , trgroup.cnt)
3202         put2output(tableconcat{
3203             "\\special{pdf:bxobj " , objname , " bbox " , bbox , "}" ,
3204             "\\unhbox\\mplibscratchbox" ,
3205             "\\special{pdf:put @resources <<" , res , ">>}" ,
3206             "\\special{pdf:exobj}" ,
3207             "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj " , objname , "}" ,
3208         })
3209     end
3210     trgroup.cnt = (trgroup.cnt or 0) + 1
3211     local objname = format("@mplibtrgr%s" , trgroup.cnt)
3212     put2output(tableconcat{
3213         "\\special{pdf:bxobj " , objname , " bbox " , bbox , "}" ,
3214         "\\unhbox\\mplibscratchbox" ,
3215         "\\special{pdf:put @resources <<" , res , ">>}" ,
3216         "\\special{pdf:exobj <<" , grattr , ">>}" ,
3217         "\\luamplibtagasgroupput{" , trgroup.name , "}" ,
3218         "\\special{pdf:uxobj " , objname , "}" ,
3219         "}\\endgroup",
3220 })
3221     token.set_macro("luamplib.group."..trgroup.name, tableconcat{
3222         "\\setbox\\mplibscratchbox\\hbox{\\hskip" , -llx , "bp\\raise" , -lly , "bp\\hbox{" ,
3223         "\\special{pdf:uxobj " , objname , "}" ,
3224         "}}\\wd\\mplibscratchbox" , urx-llx , "bp\\ht\\mplibscratchbox" , ury-lly , "bp" ,
3225         "\\box\\mplibscratchbox" ,
3226         } , "global")
3227     end
3228     trgroup.shifts[trgroup.name] = { llx , lly }
3229 end
3230 return grstate
3231 end
3232 function luamplib.registergroup (boxid, name, opts)

```

```

3233 if opts.asgroup and opts.asgroup:find"wrapped" then
3234     luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3235     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
3236     opts.asgroup = opts.asgroup:gsub("wrapped", "")
3237 end
3238 local box = texgetbox(boxid)
3239 local wd, ht, dp = node.getwhd(box)
3240 local is_mask = opts.asgroup and opts.asgroup:find"masking"
3241 local res = opts.resources or ""
3242 res = gather_resources(res)
3243 local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3244 if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
3245 if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
3246 if opts.matrix and opts.matrix:find"%a" then
3247     local t = get_mp_matrix(opts.matrix)
3248     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
3249 end
3250 local grtype = 3
3251 if opts.bbox then
3252     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3253     grtype = 2
3254 end
3255 local mpllx, mplly = get_macro'MPlLx', get_macro'MPlLy'
3256 if is_mask then
3257     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3258     t[5], t[6] = t[5]+mpllx, t[6]+mplly
3259     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
3260     mpllx, mplly = 0, 0
3261 end
3262 if opts.matrix then
3263     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3264     grtype = opts.bbox and 4 or 1
3265 end
3266 if opts.asgroup and not opts.asgroup:find"off" then
3267     local t = { isolated = false, knockout = false, masking = false }
3268     for _, v in ipairs(opts.asgroup:gsub("%s", ""):explode",+") do t[v] = true end
3269     local on
3270     if t.masking then
3271         on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorsname or "/DeviceGray"))
3272     else
3273         local cs = opts.colorsname and ("/CS%s"):format(opts.colorsname) or ""
3274         on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3275     end
3276     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3277 end
3278 local trgroup = pdfetcs.tr_group
3279 trgroup.shifts[name] = { mpllx, mplly }
3280 local whd
3281 if pdfmode then

```

```

3282 attr = tableconcat(attr) :gsub(decimals,rmzeros)
3283 local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3284 token.set_macro("luamplib.group.."name, tableconcat{
3285     "\\useboxresource ", index,
3286     }, "global")
3287 whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3288 else
3289     trgroup.cnt = (trgroup.cnt or 0) + 1
3290     local objname = format("@mplibtrgr%s", trgroup.cnt)
3291     texsprint {
3292         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3293         "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3294         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3295         "\\special{pdf:bcontent}",
3296         "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3297         "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3298         "\\special{pdf:put @resources <<", res, ">>}",
3299         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3300         "\\special{pdf:econtent}}",
3301     }
3302     token.set_macro("luamplib.group.."name, tableconcat{
3303         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
3304         "\\wd\\mplibscratchbox ", wd, "sp",
3305         "\\ht\\mplibscratchbox ", ht, "sp",
3306         "\\dp\\mplibscratchbox ", dp, "sp",
3307         "\\box\\mplibscratchbox",
3308     }, "global")
3309     whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3310 end
3311 info("w/h/d of group '%s': %s", name, whd)
3312 end
3313

```

luamplib.convert: flushing figures

```

3314 do
3315     local function stop_special_effects(fade,opaq,over)
3316         if fade then -- fading
3317             stop_pdf_code()
3318         end
3319         if opaq then -- opacity
3320             pdf_literalcode(opaq)
3321         end
3322         if over then -- color
3323             if over:find"pdf:bc" then
3324                 put2output"\\special{pdf:ec}"
3325             else
3326                 put2output"\\special{color pop}"
3327             end
3328         end

```

```
3329 end
```

```
3330
```

For parsing prescript materials.

```
3331 local function script2table(s)
3332   local t = {}
3333   for _,i in ipairs(s:explode("\13+")) do
3334     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3335     if k and v and k ~= "" and not t[k] then
3336       t[k] = v
3337     end
3338   end
3339   return t
3340 end
3341
```

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

```
3342 local function pdf_textfigure(font,size,text,width,height,depth)
3343   text = text:gsub(".",function(c)
3344     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
3345   end)
3346   put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
3347 end
3348
3349 local bend_tolerance = 131/65536
3350
3351 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3352
3353 local function pen_characteristics(object)
3354   local t = mplib.pen_info(object)
3355   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3356   divider = sx*sy - rx*ry
3357   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3358 end
3359
3360 local function concat(px, py) -- no tx, ty here
3361   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
3362 end
3363
3364 local function curved(ith,pth)
3365   local d = pth.left_x - ith.right_x
3366   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3367     abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3368     d = pth.left_y - ith.right_y
3369     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3370       abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3371       return false
3372     end
3373   end
end
```

```

3374     return true
3375 end
3376
3377 local function flushnormalpath(path,open)
3378     local pth, ith
3379     for i=1,#path do
3380         pth = path[i]
3381         if not ith then
3382             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3383         elseif curved(ith,pth) then
3384             pdf_literalcode("%f %f %f %f %f %f c",
3385                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3386         else
3387             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3388         end
3389         ith = pth
3390     end
3391     if not open then
3392         local one = path[1]
3393         if curved(pth,one) then
3394             pdf_literalcode("%f %f %f %f %f %f c",
3395                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3396         else
3397             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3398         end
3399     elseif #path == 1 then -- special case .. draw point
3400         local one = path[1]
3401         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3402     end
3403 end
3404
3405 local function flushconcatpath(path,open)
3406     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3407     local pth, ith
3408     for i=1,#path do
3409         pth = path[i]
3410         if not ith then
3411             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3412         elseif curved(ith,pth) then
3413             local a, b = concat(ith.right_x,ith.right_y)
3414             local c, d = concat(pth.left_x,pth.left_y)
3415             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3416         else
3417             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3418         end
3419         ith = pth
3420     end
3421     if not open then
3422         local one = path[1]

```

```

3423     if curved(pth,one) then
3424         local a, b = concat(pth.right_x,pth.right_y)
3425         local c, d = concat(one.left_x,one.left_y)
3426         pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3427     else
3428         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3429     end
3430 elseif #path == 1 then -- special case .. draw point
3431     local one = path[1]
3432     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3433 end
3434 end
3435

```

Finally, flush figures by inserting PDF literals.

```

3436 local function flush (result,flusher)
3437     if result then
3438         local figures = result.fig
3439         if figures then
3440             for f=1, #figures do
3441                 info("flushing figure %s",f)
3442                 local figure = figures[f]
3443                 local objects = figure:objects()
3444                 local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3445                 local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3446                 local bbox = figure:boundingbox()
3447                 local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3448                 if urx < llx then

```

luamplib silently ignores this invalid figure for those that do not contain `beginfig ... endfig`.  
(issue #70) Original code of ConT<sub>E</sub>Xt general was:

```

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

3449     else

```

For legacy behavior, insert ‘pre-fig’ T<sub>E</sub>X code here.

```

3450         if tex_code_pre_mplib[f] then
3451             put2output(tex_code_pre_mplib[f])
3452         end
3453         pdf_startfigure(fignum,llx,lly,urx,ury)
3454         start_pdf_code()
3455         if objects then
3456             local savedpath = nil
3457             local savedhtap = nil
3458             for o=1,#objects do
3459                 local object      = objects[o]
3460                 local objecttype  = object.type

```

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

```

3461         local prescript      = object.prescript
3462         prescript = prescript and script2table(prescript) -- prescript is now a table
3463         local cr_over = do_preobj_CR(object,prescript) -- color
3464         local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3465         local fading_ = do_preobj_FADE(object,prescript) -- fading
3466         local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
3467         local shading_ = do_preobj_shading(object,prescript) -- shading pattern
3468         local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3469         if prescript and prescript.mplibtexboxid then
3470             put_tex_boxes(object,prescript)
3471         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3472         elseif objecttype == "start_clip" then
3473             local evenodd = not object.istext and object.postscript == "evenodd"
3474             start_pdf_code()
3475             flushnormalpath(object.path,false)
3476             pdf_literalcode(evenodd and "W* n" or "W n")
3477         elseif objecttype == "stop_clip" then
3478             stop_pdf_code()
3479             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3480         elseif objecttype == "special" then

```

Collect T<sub>E</sub>X codes that will be executed after flushing. Legacy behavior.

```

3481         if prescript and prescript.postmplibverbtex then
3482             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtex
3483         end
3484         elseif objecttype == "text" then
3485             local ot = object.transform -- 3,4,5,6,1,2
3486             start_pdf_code()
3487             pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3488             pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3489             stop_pdf_code()
3490         elseif not trgroup and fading_ ~= "stop" then
3491             local evenodd, collect, both = false, false, false
3492             local postscript = object.postscript
3493             if not object.istext then
3494                 if postscript == "evenodd" then
3495                     evenodd = true
3496                 elseif postscript == "collect" then
3497                     collect = true
3498                 elseif postscript == "both" then
3499                     both = true
3500                 elseif postscript == "eoboth" then
3501                     evenodd = true
3502                     both = true
3503             end
3504         end
3505         if collect then
3506             if not savedpath then

```

```

3507         savedpath = { object.path or false }
3508         savedhtap = { object.htap or false }
3509     else
3510         savedpath[#savedpath+1] = object.path or false
3511         savedhtap[#savedhtap+1] = object.htap or false
3512     end
3513 else

```

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

```

3514         local ml = object.miterlimit
3515         if ml and ml ~= miterlimit then
3516             miterlimit = ml
3517             pdf_literalcode("%f M",ml)
3518         end
3519         local lj = object.linejoin
3520         if lj and lj ~= linejoin then
3521             linejoin = lj
3522             pdf_literalcode("%i j",lj)
3523         end
3524         local lc = object.linecap
3525         if lc and lc ~= linecap then
3526             linecap = lc
3527             pdf_literalcode("%i J",lc)
3528         end
3529         local dl = object.dash
3530         if dl then
3531             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3532             if d ~= dashed then
3533                 dashed = d
3534                 pdf_literalcode(dashed)
3535             end
3536         elseif dashed then
3537             pdf_literalcode("[ ] 0 d")
3538             dashed = false
3539         end
3540         local path = object.path
3541         local transformed, penwidth = false, 1
3542         local open = path and path[1].left_type and path[#path].right_type
3543         local pen = object.pen
3544         if pen then
3545             if pen.type == 'elliptical' then
3546                 transformed, penwidth = pen_characteristics(object) -- boolean, value
3547                 pdf_literalcode("%f w",penwidth)
3548                 if objecttype == 'fill' then
3549                     objecttype = 'both'
3550                 end
3551             else -- calculated by mplib itself
3552                 objecttype = 'fill'
3553             end

```

3554 end

Added : shading

```
3555 local shade_no, shade_stroke, shade_cm = do_preobj_SH(object,prescript) -- shading
3556 if shade_no then
3557     pdf_literalcode"q /Pattern cs"
3558     objecttype = false
3559 end
3560 if transformed then
3561     start_pdf_code()
3562 end
3563 if path then
3564     if savedpath then
3565         for i=1,#savedpath do
3566             local path = savedpath[i]
3567             if transformed then
3568                 flushconcatpath(path,open)
3569             else
3570                 flushnormalpath(path,open)
3571             end
3572         end
3573         savedpath = nil
3574     end
3575     if transformed then
3576         flushconcatpath(path,open)
3577     else
3578         flushnormalpath(path,open)
3579     end
3580     if objecttype == "fill" then
3581         pdf_literalcode(evenodd and "h f*" or "h f")
3582     elseif objecttype == "outline" then
3583         if both then
3584             pdf_literalcode(evenodd and "h B*" or "h B")
3585         else
3586             pdf_literalcode(open and "S" or "h S")
3587         end
3588     elseif objecttype == "both" then
3589         pdf_literalcode(evenodd and "h B*" or "h B")
3590     end
3591 end
3592 if transformed then
3593     stop_pdf_code()
3594 end
3595 local path = object.htap
```

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

```
3596 if path then
3597     if transformed then
3598         start_pdf_code()
3599     end
```

```

3600         if savedhtap then
3601             for i=1,#savedhtap do
3602                 local path = savedhtap[i]
3603                 if transformed then
3604                     flushconcatpath(path,open)
3605                 else
3606                     flushnormalpath(path,open)
3607                 end
3608             end
3609             savedhtap = nil
3610             evenodd = true
3611         end
3612         if transformed then
3613             flushconcatpath(path,open)
3614         else
3615             flushnormalpath(path,open)
3616         end
3617         if objecttype == "fill" then
3618             pdf_literalcode(evenodd and "h f*" or "h f")
3619         elseif objecttype == "outline" then
3620             pdf_literalcode(open and "S" or "h S")
3621         elseif objecttype == "both" then
3622             pdf_literalcode(evenodd and "h B*" or "h B")
3623         end
3624         if transformed then
3625             stop_pdf_code()
3626         end
3627     end

```

Added to ConT<sub>E</sub>Xt general: post-object colors and shading stuff. Beware q ... Q scope.

```

3628         if shade_no then -- shading
3629             pdf_literalcode("W%s %s %s/MPLibSh%s sh Q",
3630                 evenodd and "*" or "",
3631                 shade_stroke and "s" or "n",
3632                 shade_cm and shade_cm.." cm " or "",
3633                 shade_no)
3634         end
3635     end
3636 end
3637 if fading_ == "start" then
3638     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}
3639 elseif trgroup == "start" then
3640     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
3641 elseif fading_ == "stop" then
3642     local se = pdfetcs.fading.specialeffects
3643     if se then stop_special_effects(se[1], se[2], se[3]) end
3644 elseif trgroup == "stop" then
3645     local se = pdfetcs.tr_group.specialeffects
3646     if se then stop_special_effects(se[1], se[2], se[3]) end

```

```

3647         else
3648             stop_special_effects(fading_, tr_opaq, cr_over)
3649         end
3650         if fading_ or trgroup then -- extgs resetted
3651             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3652         end
3653     end
3654 end
3655 stop_pdf_code()
3656 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtex code.

```

3657         for _,v in ipairs(figcontents) do
3658             if type(v) == "table" then
3659                 texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3660             else
3661                 texsprint(v)
3662             end
3663         end
3664         if #figcontents.post > 0 then texsprint(figcontents.post) end
3665         figcontents = { post = { } }
3666     end
3667 end
3668 end
3669 end
3670 end
3671
3672 function luamplib.convert (result, flusher)
3673     flush(result, flusher)
3674     return true -- done
3675 end
3676 end
3677
3678 function luamplib.colorconverter (cr)
3679     local n = #cr
3680     if n == 4 then
3681         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3682         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3683     elseif n == 3 then
3684         local r, g, b = cr[1], cr[2], cr[3]
3685         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3686     else
3687         local s = cr[1]
3688         return format("%.3f g %.3f G",s,s), "0 g 0 G"
3689     end
3690 end

```

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

First we need to load some packages.

```
3691 \ifcsname ProvidesPackage\endcsname
```

We need L<sup>A</sup>T<sub>E</sub>X 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.

```
3692 \NeedsTeXFormat{LaTeX2e}
3693 \ProvidesPackage{luamplib}
3694 [2026/06/19 v2.42.2 mplib package for LuaTeX]
3695 \fi
3696 \ifdefined\newluafunction\else
3697 \input ltluatex
3698 \fi
```

In DVI mode, a new XObject (mppattern, mplibgroup) must be encapsulated in an `\hbox`. But this should not affect typesetting. So we use Hook mechanism provided by L<sup>A</sup>T<sub>E</sub>X kernel. In Plain, `atbegshi.sty` is loaded.

```
3699 \ifnum\outputmode=0
3700 \ifdefined\AddToHookNext
3701 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3702 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3703 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3704 \else
3705 \input atbegshi.sty
3706 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3707 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3708 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3709 \fi
3710 \fi
```

Loading of lua code.

```
3711 \directlua{require("luamplib")}
```

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

```
3712 \ifx\pdfoutput\undefined
3713 \let\pdfoutput\outputmode
3714 \fi
3715 \ifx\pdfliteral\undefined
3716 \protected\def\pdfliteral{\pdfextension literal}
3717 \fi
```

Set the format for METAPOST.

```
3718 \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.

```
3719 \ifnum\pdfoutput>0
3720 \let\mplibtoPDF\pdfliteral
3721 \else
```

```

3722 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3723 \ifcsname PackageInfo\endcsname
3724   \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3725 \else
3726   \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3727 \fi
3728 \fi

```

To make mplibcode typeset always in horizontal mode.

```

3729 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3730 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3731 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

3732 \def\mplibsetupcatcodes{%
3733   %catcode`\{=12 %catcode`\}=12
3734   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3735   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
3736 }

```

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

```

3737 \ifnum\outputmode>0 %
3738   \def\mplibputtextbox#1#2#3#4{%
3739     \pdfextension save\relax
3740     \vbox to 0pt{\vss
3741       \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
3742       \kern#3bp}%
3743     \pdfextension restore\relax}
3744 \else
3745   \def\mplibputtextbox#1#2#3#4{%
3746     \special{pdf:btrans matrix #4 #2 #3}%
3747     \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
3748     \special{pdf:etrans}}
3749 \fi

```

use Transparency Group

```

3750 \protected\def\usemplibgroup#1#{\usemplibgroupmain}
3751 \def\usemplibgroupmain#1{%
3752   \prependtomplibbox\hbox dir TLT\bgroup
3753   \csname luamplib.group.#1\endcsname
3754   \egroup
3755 }
3756 \protected\def\mplibgroup#1{%
3757   \begingroup
3758   \def\MPllx{0}\def\MPlly{0}%
3759   \def\mplibgroupname{#1}%
3760   \mplibgroupgetnexttok
3761 }
3762 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3763 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok= }

```

```

3764 \def\mplibgroupbranch{%
3765   \ifx [\nexttok
3766     \expandafter\mplibgroupopts
3767   \else
3768     \ifx\mplibsptoken\nexttok
3769       \expandafter\expandafter\expandafter\mplibgroupskipspace
3770     \else
3771       \let\mplibgroupoptions\empty
3772     \expandafter\expandafter\expandafter\mplibgroupmain
3773   \fi
3774 \fi
3775 }
3776 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3777 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3778 \protected\def\endmplibgroup{\egroup
3779   \directlua{ luamplib.registergroup(
3780     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3781   )}%
3782 \endgroup
3783 }

```

## Patterns

```

3784 {\def\:{\global\let\mplibsptoken= } \: }
3785 \protected\def\mppattern#1{%
3786   \begingroup
3787   \def\MPllx{0}\def\MPlly{0}%
3788   \def\mplibpatternname{#1}%
3789   \mplibpatterngetnexttok
3790 }
3791 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3792 \def\mplibpatternskipspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3793 \def\mplibpatternbranch{%
3794   \ifx [\nexttok
3795     \expandafter\mplibpatternopts
3796   \else
3797     \ifx\mplibsptoken\nexttok
3798       \expandafter\expandafter\expandafter\mplibpatternskipspace
3799     \else
3800       \let\mplibpatternoptions\empty
3801     \expandafter\expandafter\expandafter\mplibpatternmain
3802   \fi
3803 \fi
3804 }
3805 \def\mplibpatternopts[#1]{%
3806   \def\mplibpatternoptions{#1}%
3807   \mplibpatternmain
3808 }
3809 \def\mplibpatternmain{%
3810   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces

```

```

3811 }
3812 \protected\def\endmppattern{%
3813   \egroup
3814   \directlua{ luamplib.registerpattern(
3815     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3816   )}%
3817   \endgroup
3818 }

```

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

```

3819 \def\mpfiginstancename{@mpfig}
3820 \protected\def\mpfig{%
3821   \begingroup
3822   \futurelet\nexttok\mplibmpfigbranch
3823 }
3824 \def\mplibmpfigbranch{%
3825   \ifx *\nexttok
3826     \expandafter\mplibprempfig
3827   \else
3828     \ifx [\nexttok
3829       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3830     \else
3831       \expandafter\expandafter\expandafter\mplibmainmpfig
3832     \fi
3833   \fi
3834 }
3835 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
3836 \def\mplibmainmpfig{%
3837   \begingroup
3838   \mplibsetupcatcodes
3839   \mplibdomainmpfig
3840 }
3841 \long\def\mplibdomainmpfig#1\endmpfig{%
3842   \endgroup
3843   \directlua{
3844     local legacy = luamplib.legacyverbatimimtex
3845     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3846     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3847     luamplib.legacyverbatimimtex = false
3848     luamplib.everymplib["\mpfiginstancename"] = ""
3849     luamplib.everyendmplib["\mpfiginstancename"] = ""
3850     luamplib.process_mplibcode(
3851       "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===".." "..everyendmpfig.." endfig;",
3852       "\mpfiginstancename")
3853     luamplib.legacyverbatimimtex = legacy
3854     luamplib.everymplib["\mpfiginstancename"] = everympfig
3855     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3856   }%
3857   \endgroup

```

```

3858 }
3859 \def\mplibprempfig#1{%
3860   \begingroup
3861   \mplibsetupcatcodes
3862   \mplibdoprempfig
3863 }
3864 \long\def\mplibdoprempfig#1\endmpfig{%
3865   \endgroup
3866   \directlua{
3867     local legacy = luamplib.legacyverbatim
3868     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3869     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3870     luamplib.legacyverbatim = false
3871     luamplib.everymplib["\mpfiginstancename"] = ""
3872     luamplib.everyendmplib["\mpfiginstancename"] = ""
3873     luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
3874     luamplib.legacyverbatim = legacy
3875     luamplib.everymplib["\mpfiginstancename"] = everympfig
3876     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3877   }%
3878   \endgroup
3879 }
3880 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3881 \unless\ifcsname ver@luamplib.sty\endcsname
3882   \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3883   \protected\def\mplibcode{%
3884     \begingroup
3885     \futurelet\nexttok\mplibcodebranch
3886   }
3887   \def\mplibcodebranch{%
3888     \ifx [\nexttok
3889       \expandafter\mplibcodegetinstancename
3890     \else
3891       \global\let\currentmpinstancename\empty
3892       \expandafter\mplibcodeindeed
3893     \fi
3894   }
3895   \def\mplibcodeindeed{%
3896     \begingroup
3897     \mplibsetupcatcodes
3898     \mplibdocode
3899   }
3900   \long\def\mplibdocode#1\endmplibcode{%
3901     \endgroup
3902     \directlua{luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\currentmpinstancename")}%
3903   \endgroup
3904 }

```

```

3905 \protected\def\endmplibcode{\endmplibcode}
3906 \else

```

The L<sup>A</sup>T<sub>E</sub>X-specific part: a new environment.

```

3907 \newenvironment{mplibcode}[1][{}]{%
3908   \xdef\currentmpinstancename{#1}%
3909   \mplibtmptoks{}\ltxdomplibcode
3910 }{}
3911 \def\ltxdomplibcode{%
3912   \begingroup
3913   \mplibsetupcatcodes
3914   \ltxdomplibcodeindeed
3915 }
3916 \def\mplib@mplibcode{mplibcode}
3917 \long\def\ltxdomplibcodeindeed#1\end#2{%
3918   \endgroup
3919   \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3920   \def\mplibtemp@a{#2}%
3921   \ifx\mplib@mplibcode\mplibtemp@a
3922     \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]==],"\currentmpinstancename")}%
3923     \end{mplibcode}%
3924   \else
3925     \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3926     \expandafter\ltxdomplibcode
3927   \fi
3928 }
3929 \fi

```

User settings.

```

3930 \def\mplibshowlog#1{\directlua{
3931   local s = string.lower("#1")
3932   if s == "enable" or s == "true" or s == "yes" then
3933     luamplib.showlog = true
3934   else
3935     luamplib.showlog = false
3936   end
3937 }}
3938 \def\mpliblegacybehavior#1{\directlua{
3939   local s = string.lower("#1")
3940   if s == "enable" or s == "true" or s == "yes" then
3941     luamplib.legacyverbatim = true
3942   else
3943     luamplib.legacyverbatim = false
3944   end
3945 }}
3946 \def\mplibverbatim#1{\directlua{
3947   local s = string.lower("#1")
3948   if s == "enable" or s == "true" or s == "yes" then
3949     luamplib.verbatiminput = true
3950   else

```

```

3951     luamplib.verbatiminput = false
3952 end
3953 }}
3954 \newtoks\mplibmptoks

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

3955 \ifcsname ver@luamplib.sty\endcsname
3956 \protected\def\everymplib{%
3957   \begingroup
3958   \mplibsetupcatcodes
3959   \mplibdoeverymplib
3960 }
3961 \protected\def\everyendmplib{%
3962   \begingroup
3963   \mplibsetupcatcodes
3964   \mplibdoeveryendmplib
3965 }
3966 \newcommand\mplibdoeverymplib[2][]{%
3967   \endgroup
3968   \directlua{
3969     luamplib.everymplib["#1"] = [===[\unexpanded{#2}]==]
3970   }%
3971 }
3972 \newcommand\mplibdoeveryendmplib[2][]{%
3973   \endgroup
3974   \directlua{
3975     luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]==]
3976   }%
3977 }
3978 \else
3979 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3980 \protected\def\everymplib#1{%
3981   \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3982   \begingroup
3983   \mplibsetupcatcodes
3984   \mplibdoeverymplib
3985 }
3986 \long\def\mplibdoeverymplib#1{%
3987   \endgroup
3988   \directlua{
3989     luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]==]
3990   }%
3991 }
3992 \protected\def\everyendmplib#1{%
3993   \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3994   \begingroup
3995   \mplibsetupcatcodes
3996   \mplibdoeveryendmplib
3997 }

```

```

3998 \long\def\mplibdoeveryendmplib#1{%
3999   \endgroup
4000   \directlua{
4001     luampLib.everyendmplib["\currentmpinstancename"] = [==[\unexpanded{#1}]==]
4002   }%
4003 }
4004 \fi

```

TeX macros for dimen/color

```

4005 \def\mpdim#1{ runscript("luampLibdimen{#1}") }
4006 \def\mpcolor#1#\domplibcolor{#1}}
4007 \def\domplibcolor#1#2{ runscript("luampLibcolor{#1{#2}}") }

```

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

```

4008 \def\mplibnumbersystem#1{\directlua{
4009   local t = "#1"
4010   if t == "binary" then t = "decimal" end
4011   luampLib.numbersystem = t
4012 }}

```

Settings for .mp cache files.

```

4013 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4014 \def\mplibdomakenocache#1,{%
4015   \ifx\empty#1\empty
4016     \expandafter\mplibdomakenocache
4017   \else
4018     \ifx\stop#1\else
4019       \directlua{luampLib.noneedtoreplace["#1.mp"]=true}%
4020       \expandafter\expandafter\expandafter\mplibdomakenocache
4021     \fi
4022   \fi
4023 }
4024 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4025 \def\mplibdocancelnocache#1,{%
4026   \ifx\empty#1\empty
4027     \expandafter\mplibdocancelnocache
4028   \else
4029     \ifx\stop#1\else
4030       \directlua{luampLib.noneedtoreplace["#1.mp"]=false}%
4031       \expandafter\expandafter\expandafter\mplibdocancelnocache
4032     \fi
4033   \fi
4034 }
4035 \def\mplibcachedir#1{\directlua{luampLib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4036 \def\mplibtexttextlabel#1{\directlua{
4037   local s = string.lower("#1")
4038   if s == "enable" or s == "true" or s == "yes" then
4039     luampLib.texttextlabel = true

```

```

4040     else
4041         luamplib.texttextlabel = false
4042     end
4043 }}
4044 \def\mplibcodeinherit#1{\directlua{
4045     local s = string.lower("#1")
4046     if s == "enable" or s == "true" or s == "yes" then
4047         luamplib.codeinherit = true
4048     else
4049         luamplib.codeinherit = false
4050     end
4051 }}
4052 \def\mplibglobaltexttext#1{\directlua{
4053     local s = string.lower("#1")
4054     if s == "enable" or s == "true" or s == "yes" then
4055         luamplib.globaltexttext = true
4056     else
4057         luamplib.globaltexttext = false
4058     end
4059 }}

```

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

We use a dedicated scratchbox.

```

4060 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4061 \def\mplibstarttoPDF#1#2#3#4{%
4062     \prependtomplibbox
4063     \hbox dir TLT\bgroup
4064     \xdef\MPllx{#1}\xdef\MPlly{#2}%
4065     \xdef\MPurx{#3}\xdef\MPury{#4}%
4066     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4067     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4068     \parskip0pt%
4069     \leftskip0pt%
4070     \parindent0pt%
4071     \everypar{}%
4072     \setbox\mplibscratchbox\vbox\bgroup
4073     \noindent
4074 }
4075 \def\mplibstoptoPDF{%
4076     \par
4077     \egroup %
4078     \setbox\mplibscratchbox\hbox %
4079         {\hskip-\MPllx bp%
4080          \raise-\MPlly bp%
4081          \box\mplibscratchbox}%
4082     \setbox\mplibscratchbox\vbox to \MPheight
4083         {\vfill
4084          \hsize\MPwidth

```

```

4085 \wd\mplibscratchbox0pt%
4086 \ht\mplibscratchbox0pt%
4087 \dp\mplibscratchbox0pt%
4088 \box\mplibscratchbox}%
4089 \wd\mplibscratchbox\MPwidth
4090 \ht\mplibscratchbox\MPheight
4091 \box\mplibscratchbox
4092 \egroup
4093 }

```

Text items have a special handler.

```

4094 \def\mplibtexttext#1#2#3#4#5{%
4095 \begingroup
4096 \setbox\mplibscratchbox\hbox
4097 {\font\temp=#1 at #2bp%
4098 \temp
4099 #3}%
4100 \setbox\mplibscratchbox\hbox
4101 {\hskip#4 bp%
4102 \raise#5 bp%
4103 \box\mplibscratchbox}%
4104 \wd\mplibscratchbox0pt%
4105 \ht\mplibscratchbox0pt%
4106 \dp\mplibscratchbox0pt%
4107 \box\mplibscratchbox
4108 \endgroup
4109 }

```

Input luamplib.cfg when it exists.

```

4110 \openin0=luamplib.cfg
4111 \ifeof0 \else
4112 \closein0
4113 \input luamplib.cfg
4114 \fi

```

Code for tagpdf

```

4115 \def\luamplibtagtextboxset#1#2{#2}
4116 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4117 \let\luamplibtagasgroupset\relax
4118 \let\luamplibtagasgroupput\luamplibtagtextboxset
4119 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4120 \ifcsname ver@tagpdf.sty\endcsname \else
4121 \ExplSyntaxOn
4122 \keys_define:nn{luamplib/tagging}
4123 {
4124 ,alt .code:n = { }
4125 ,actualtext .code:n = { }
4126 ,artifact .code:n = { }
4127 ,text .code:n = { }
4128 ,off .code:n = { }

```

```

4129 ,tag .code:n = { }
4130 ,adjust-BBox .code:n = { }
4131 ,tagging-setup .code:n = { }
4132 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4133 ,instancename .meta:n = { instance = {#1} }
4134 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4135 }
4136 \RenewDocumentCommand\mplibcode{0{}}
4137 {
4138   \tl_gclear:N \currentmpinstancename
4139   \keys_set:ne{luamplib/tagging}{#1}
4140   \mplibmptoks{}\ltxdomplibcode
4141 }
4142 \cs_set_eq:NN \mplibaltext \use_none:n
4143 \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  $\text{\LaTeX}$ . 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  $\text{\LaTeX}$ 's `\everypar` should have done.

```

4144 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4145 \ExplSyntaxOff
4146 \endinput\fi
4147 \ExplSyntaxOn
4148 \tl_new:N \l__luamplib_tag_envname_tl
4149 \tl_new:N \l__luamplib_tag_alt_tl
4150 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4151 \tl_new:N \l__luamplib_tag_actual_tl
4152 \tl_new:N \l__luamplib_tag_struct_tl
4153 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4154 \bool_new:N \l__luamplib_tag_usetext_bool
4155 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4156 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4157 \tl_new:N \l__luamplib_tag_bbox_draw_tl
4158 \tl_new:N \l__luamplib_BBox_llx_tl
4159 \tl_new:N \l__luamplib_BBox_lly_tl
4160 \tl_new:N \l__luamplib_BBox_urx_tl
4161 \tl_new:N \l__luamplib_BBox_ury_tl
4162 \msg_new:nnn {luamplib}{figure-text-reuse}
4163 {
4164   tex-text~box~#1~probably~is~incorrectly~tagged.~
4165   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4166   Check~the~resulting~PDF.
4167 }
4168 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4169 {
4170   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4171   Using~mplibgroup~with~text~mode~is~not~recommended.~

```

```

4172 Check~the~resulting~PDF.
4173 }
4174 \msg_new:nnn{luamplib}{alt-text-missing}
4175 {
4176   Alternate~text~for~#1~is~missing.~
4177   Using~the~default~value~'#2'~instead.
4178 }

```

Sockets for tex-text boxes.

```

4179 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4180 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4181 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4182 {

```

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.

```

4183 \bool_if:NTF \l__luamplib_tag_usetext_bool
4184 {
4185   \tag_mc_end_push:
4186   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4187   \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.

```

4188   \tag_mc_begin:n{tag=text}
4189   #2
4190   \tag_mc_end:
4191   \tag_struct_end:
4192   \tag_mc_begin_pop:n{ }
4193 }
4194 {
4195   \tag_suspend:n{\luamplibtagtextboxset}
4196   #2
4197   \tag_resume:n{\luamplibtagtextboxset}
4198 }
4199 }
4200 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4201 {
4202   \bool_lazy_and:nnTF
4203   { \l__luamplib_tag_usetext_bool }
4204   { \cs_if_free_p:c {luamplib.notaggedbox.#1} }
4205   {
4206     \tag_resume:n{\mplibputtextbox}
4207     \tag_mc_end:
4208     \cs_if_exist:cTF {luamplib.taggedbox.#1}
4209     {
4210       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4211       #2
4212       \cs_undefine:c {luamplib.taggedbox.#1}
4213     }
4214   {

```

```

4215 \msg_warning:nn{luamplib}{figure-text-reuse}{#1}
4216 \tag_mc_begin:n{}
4217 \int_set:Nn \l_tmpa_int {#1}
4218 \tag_mc_reset_box:N \l_tmpa_int
4219 #2
4220 \tag_mc_end:
4221 }
4222 \tag_mc_begin:n{artifact}
4223 }
4224 {
4225 \int_set:Nn \l_tmpa_int {#1}
4226 \tag_mc_reset_box:N \l_tmpa_int
4227 #2
4228 }
4229 }
4230 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4231 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4232 \cs_set_nopar:Npn \luamplibtagtextboxset
4233 {
4234 \tag_socket_use:nnn{luamplib/texttext/set}
4235 }

```

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

```

4236 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4237 {
4238 \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
4239 \bool_set_false:N \l__luamplib_tag_usetext_bool
4240 \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
4241 \cs_get_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4242 \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4243 }
4244 \sys_if_output_pdf:TF
4245 {
4246 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4247 {
4248 \pdfextension save\relax
4249 \vbox to 0pt{\vss
4250 \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4251 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4252 \kern #3bp}
4253 \pdfextension restore\relax
4254 }
4255 }
4256 {
4257 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4258 {
4259 \special{pdf:btrans~matrix~#4~#2~#3}
4260 \vbox to 0pt{\vss\hbox bdir0 to 0pt{

```

```

4261 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4262 \special{pdf:etrans}
4263 }
4264 }

```

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

```

4265 \cs_set_nopar:Npn \luamplibtagasgroupset
4266 {
4267   \bool_set_false:N \l__luamplib_tag_usetext_bool
4268 }
4269 \cs_set_nopar:Npn \luamplibtagasgroupput
4270 {
4271   \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4272   \tag_socket_use:nnn{luamplib/mpplibgroup/put}
4273 }

```

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

```

4274 \socket_new:nn{tagsupport/luamplib/mpplibgroup/put}{2}
4275 \socket_new_plug:nnn{tagsupport/luamplib/mpplibgroup/put}{default}
4276 {
4277   \cs_if_free:cT {luamplib.mpplibgroup.text.#1}
4278   {
4279     \msg_warning:nnn {luamplib} {mpplibgroup-text-mode} {#1}
4280     \cs_gset_nopar:cpn {luamplib.mpplibgroup.text.#1} {#1}
4281   }
4282   \tag_mc_end:
4283   \tag_mc_begin:n{tag=text}
4284   #2
4285   \tag_mc_end:
4286   \tag_mc_begin:n{artifact}
4287 }
4288 \socket_assign_plug:nn{tagsupport/luamplib/mpplibgroup/put}{default}

```

A macro for BBox attribute

```

4289 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4290 {
4291   \tl_set:Ne \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4292   \tex_savepos:D
4293   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4294   \tl_set:Ne \l__luamplib_BBox_llx_tl
4295   { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4296   \tl_set:Ne \l__luamplib_BBox_lly_tl
4297   { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4298   \tl_set:Ne \l__luamplib_BBox_urx_tl
4299   { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4300   \tl_set:Ne \l__luamplib_BBox_ury_tl
4301   { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4302   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4303   {

```

```

4304 \int_zero:N \l_tmpa_int
4305 \tl_map_inline:nn
4306 {
4307   \l__luamplib_BBox_llx_tl
4308   \l__luamplib_BBox_lly_tl
4309   \l__luamplib_BBox_urx_tl
4310   \l__luamplib_BBox_ury_tl
4311 }
4312 {
4313   \int_incr:N \l_tmpa_int
4314   \tl_set:Ne ##1
4315   {
4316     \fp_eval:n
4317     {
4318       ##1
4319       +
4320       \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4321     }
4322   }
4323 }
4324 }
4325 \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4326 {
4327   /O /Layout /BBox [
4328     \l__luamplib_BBox_llx_tl\c_space_tl
4329     \l__luamplib_BBox_lly_tl\c_space_tl
4330     \l__luamplib_BBox_urx_tl\c_space_tl
4331     \l__luamplib_BBox_ury_tl
4332   ]
4333 }
4334 \bool_if:NT \l__tag_graphic_debug_bool
4335 {
4336   \iow_log:e
4337   {
4338     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4339     \l__luamplib_BBox_llx_tl\c_space_tl
4340     \l__luamplib_BBox_lly_tl\c_space_tl
4341     \l__luamplib_BBox_urx_tl\c_space_tl
4342     \l__luamplib_BBox_ury_tl
4343   }
4344   \sys_if_output_pdf:TF
4345   {
4346     \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
4347     {
4348       \pdfextension save\relax
4349       \opacity_select:n{0.5} \color_select:n{red}
4350       \pdfextension literal~text
4351       {
4352         \l__luamplib_BBox_llx_tl\c_space_tl

```

```

4353     \l__luamplib_BBox_lly_tl\c_space_tl
4354     \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4355     \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4356     re~f
4357   }
4358   \pdfextension restore\relax
4359 }
4360 }
4361 {
4362   \tl_set:Nc \l__luamplib_tag_bbox_draw_tl
4363   {
4364     \special{pdf:bcontent}
4365     \opacity_select:n{0.5} \color_select:n{red}
4366     \special{pdf:code~
4367       1~0~0~1~
4368       -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4369       -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4370       cm
4371     }
4372     \special{pdf:code~
4373       \l__luamplib_BBox_llx_tl\c_space_tl
4374       \l__luamplib_BBox_lly_tl\c_space_tl
4375       \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4376       \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4377       re~f
4378     }
4379     \special{pdf:econtent}
4380   }
4381 }
4382 }
4383 }

```

### Sockets for main process

```

4384 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4385 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4386 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4387 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4388 {
4389   \tag_mc_end_push:
4390   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4391   {
4392     \tl_if_empty:eTF{#1}
4393     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4394     { \tl_set:Nc \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4395     \msg_warning:nnVV{luamplib}{alt-text-missing}
4396     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4397   }
4398   \tag_struct_begin:n
4399   {

```

```

4400     tag=\l__luamplib_tag_struct_tl,
4401     alt=\l__luamplib_tag_alt_tl,
4402   }
4403   \tag_mc_begin:n{}
4404 }
4405 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4406 {
4407   \__luamplib_tag_bbox_attribute:n {#1}
4408   #2
4409   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4410   \tag_mc_end:
4411   \tag_struct_end:
4412   \tag_mc_begin_pop:n{}
4413 }
4414 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4415 {
4416   \tag_mc_end_push:
4417   \tag_struct_begin:n
4418   {
4419     tag=Span,
4420     actualtext=\l__luamplib_tag_actual_tl,
4421   }
4422   \tag_mc_begin:n{}
4423 }
4424 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4425 {
4426   #2
4427   \tag_mc_end:
4428   \tag_struct_end:
4429   \tag_mc_begin_pop:n{}
4430 }
4431 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4432 {
4433   \tag_mc_end_push:
4434   \tag_mc_begin:n{artifact}
4435 }
4436 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4437 {
4438   #2
4439   \tag_mc_end:
4440   \tag_mc_begin_pop:n{}
4441 }

```

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

```

4442 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4443 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4444 {
4445   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}

```

```

4446 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4447 }
4448 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4449 {
4450 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4451 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4452 \prependtomplibbox \mplibnoforcehmode
4453 \mode_if_vertical:T { \noindent \aftergroup\par }
4454 }
4455 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4456 {
4457 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4458 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4459 }
4460 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4461 {
4462 \bool_set_true:N \l__luamplib_tag_usetext_bool
4463 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4464 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4465 \prependtomplibbox \mplibnoforcehmode
4466 \mode_if_vertical:T { \noindent \aftergroup\par }
4467 }
4468 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4469 {
4470 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4471 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4472 }
4473 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

### Key-value options

```

4474 \keys_define:nn{luamplib/tagging}
4475 {
4476 ,alt .code:n =
4477 {
4478 \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4479 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4480 }
4481 ,actualtext .code:n =
4482 {
4483 \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4484 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4485 }
4486 ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4487 ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4488 ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4489 ,tag .code:n =
4490 {
4491 \str_case:nnF {#1}

```

```

4492 {
4493   {false} { \keys_set:nn {luamplib/tagging} {off} }
4494   {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4495 }
4496 {
4497   \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4498   \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4499 }
4500 }
4501 ,adjust-BBox .code:n =
4502 {
4503   \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4504   \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4505 }
4506 ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4507 }
4508 \keys_define:nn {luamplib/instance}
4509 {
4510   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4511   ,instancename .meta:n = { instance = {#1} }
4512   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4513 }

```

Redefine our macros

```

4514 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4515 {
4516   \prependtomplibbox
4517   \hbox dir~TLT\bgroup
4518     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4519     \xdef\MPllx{#1}\xdef\MPlly{#2}%
4520     \xdef\MPurx{#3}\xdef\MPury{#4}%
4521     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4522     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4523     \parskip0pt
4524     \leftskip0pt
4525     \parindent0pt
4526     \everypar{}%
4527     \setbox\mplibscratchbox\vbox\bgroup
4528       \tag_suspend:n{\mplibstarttoPDF}
4529       \noindent
4530 }
4531 \cs_set_nopar:Npn \mplibstoptoPDF
4532 {
4533   \par
4534   \egroup
4535   \setbox\mplibscratchbox\hbox
4536     {\hskip-\MPllx bp
4537     \raise-\MPlly bp
4538     \box\mplibscratchbox}%

```

```

4539 \setbox\mplibscratchbox\ vbox to \MPheight
4540 {\vfill
4541 \hsize\MPwidth
4542 \wd\mplibscratchbox\0pt
4543 \ht\mplibscratchbox\0pt
4544 \dp\mplibscratchbox\0pt
4545 \box\mplibscratchbox}%
4546 \wd\mplibscratchbox\MPwidth
4547 \ht\mplibscratchbox\MPheight
4548 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4549 \egroup
4550 }
4551 \RenewDocumentCommand\mplibcode{0{}}
4552 {
4553 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4554 \tl_gclear:N \currentmpinstancename
4555 \keys_set:known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4556 \keys_set:nV {luamplib/instance} \l_tmpa_tl
4557 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4558 \tag_socket_use:n{luamplib/figure/init}
4559 \mplibtmptoks{}\ltxdomplibcode
4560 }
4561 \RenewDocumentCommand\mpfig{s 0{}}
4562 {
4563 \beginpgroup
4564 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4565 \keys_set:known:ne {luamplib/tagging} {#2}
4566 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4567 \tag_socket_use:n{luamplib/figure/init}
4568 \IfBooleanTF{#1} { \mplibprempfig * }
4569 { \mplibmainmpfig }
4570 }
4571 \RenewDocumentCommand\usemplibgroup{0{ } m}
4572 {
4573 \beginpgroup
4574 \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4575 \keys_set:known:ne {luamplib/tagging} {#1}
4576 \tag_socket_use:n{luamplib/figure/init}
4577 \prependtomplibbox\hbox dir~TLT\bgroup
4578 \tag_socket_use:nn{luamplib/figure/begin}{#2}
4579 \setbox\mplibscratchbox\hbox\bgroup
4580 \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
4581 \tag_socket_use:nnn{luamplib/mpfiggroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4582 \egroup
4583 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4584 \endpgroup
4585 \endgroup
4586 }

```

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

well.

```
4587 \cs_new_nopar:Npn \mplibalttext #1
4588 {
4589   \tl_set:Nc \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4590 }
4591 \cs_new_nopar:Npn \mplibactualtext #1
4592 {
4593   \tl_set:Nc \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4594 }
4595 \ExplSyntaxOff
```

That's all folks!

## 3 The GNU GPL License v2

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

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

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

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

**Preamble**

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

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

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

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

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

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

**TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**NO WARRANTY**

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

**END OF TERMS AND CONDITIONS**

**Appendix: How to Apply These Terms to Your New Programs**

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

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

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

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

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

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

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

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

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

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

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

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

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

This General Public License does not permit incorporating your program into proprietary programs. If your program is a 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.