

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/05/12 V2.41.2

Abstract

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

Contents

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

| | | |
|----------|---|------------|
| 1.2.5 | <code>withmplibopacities</code> | 12 |
| 1.2.6 | <code>withshadingmethod</code> | 13 |
| 1.2.7 | <code>withfademethod</code> | 15 |
| 1.2.8 | <code>mplibgraphicstext</code> | 16 |
| 1.2.9 | <code>mplibglyph</code> | 16 |
| 1.2.10 | <code>mplibdrawglyph</code> , and its friends | 17 |
| 1.2.11 | <code>mpliboutlinetext</code> | 17 |
| 1.2.12 | <code>\mppattern</code> , <code>withmppattern</code> | 18 |
| 1.2.13 | <code>asgroup</code> | 20 |
| 1.2.14 | <code>\mplibgroup</code> | 23 |
| 1.2.15 | <code>withmaskinggroup</code> | 24 |
| 1.2.16 | <code>mpliblength</code> , <code>mplibuclength</code> | 25 |
| 1.2.17 | <code>mplibsubstring</code> , <code>mplibucsubstring</code> | 25 |
| 1.3 | <code>Lua</code> | 26 |
| 1.3.1 | <code>runscript</code> | 26 |
| 1.3.2 | <code>luamplib.instances</code> | 26 |
| 1.3.3 | <code>luamplib.process_mplibcode</code> | 27 |
| 1.3.4 | <code>luamplib.registerpattern</code> | 27 |
| 1.3.5 | <code>luamplib.registergroup</code> | 28 |
| 2 | Implementation | 28 |
| 2.1 | <code>Lua module</code> | 28 |
| 2.2 | <code>TeXpackage</code> | 100 |
| 3 | The GNU GPL License v2 | 121 |

1 Documentation

This package aims at providing a simple way to typeset directly METAPOST code in a document with Lua \TeX . Lua \TeX is built with the Lua `mplib` library, that runs METAPOST code. This package is basically a wrapper for the Lua `mplib` functions and some \TeX functions to have the output of the `mplib` functions in the PDF.

Using this package is easy: in Plain, type your METAPOST code between the macros `\mplibcode` and `\endmplibcode`, and in \LaTeX in the `mplibcode` environment.

The resulting METAPOST figures are put in a \TeX hbox with dimensions adjusted to the METAPOST code.

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

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

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

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

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

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

1.1 \TeX

1.1.1 `\mplibforcehmode`

When this macro is declared, every `METAPOST` figure box will be typeset in horizontal mode, so that `\centering`, `\raggedleft` etc. will have effects. `\mplibnoforcehmode`, being default for backward compatibility, reverts this setting.¹

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` $\langle format name \rangle$.

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 § 1.2.11). You can try other effects as well, though we did not fully tested their proper functioning.

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

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

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

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

From v2.36, shading is available with *plain* format as well with extended functionality. See below § 1.2.6.

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 Foxit Editor, cannot properly render the isolated or knockout effect.

Transparency group is available with *plain* format as well with extended functionality. See below § 1.2.13.

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}`² is declared, log messages returned by the METAPOST process will be printed to the `.log` file. This is the T_EX side interface for `luamplib.showlog`.

1.1.6 `\mpliblegacybehavior{enable|disable}`

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

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

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

³But the recommended way to reuse a figure is using `\mplibgroup` command. See below § 1.2.14.

```

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

```


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

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

```

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

```



diameter: 22.62764bp.

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

```

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

```

ABC DEF GHI

1.1.7 `\mplibtexttextlabel{enable|disable}`

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

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

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

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

1.1.8 `\mplibcodeinherit{enable|disable}`

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

1.1.9 `\mplibglobaltexttext{enable|disable}`

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

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



1.1.10 Separate METAPOST instances

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

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

The behavior is as follows.

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

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

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

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

1.1.11 `\mplibverbatim{enable|disable}`

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

1.1.12 `\mpdim{...}`

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

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



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

With `\mpcolor` command, color names or expressions of `color`, `xcolor` and `l3color` module/packages can be used in the `mplibcode` environment (after `withcolor` command, in principle). See the example above at § 1.1.12. The optional `[...]` denotes the option of `xcolor`'s `\color` command. For spot colors, `l3color` 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 \LaTeX) and `\mplibcode ... \endmplibcode` (for Plain), we also provide unexpandable \TeX macros `\mpfig ... \endmpfig` and its starred version `\mpfig* ... \endmpfig` to save typing toil. The former is roughly the same as follows:

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

and the starred version is roughly the same as follows:

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

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

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

Box 1

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

1.1.15 About cache files

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

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

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

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

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

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

1.1.16 About figure box metric

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

1.1.17 luamplib.cfg

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

1.1.18 Tagged PDF

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

alt= $\langle text \rangle$ starts a Figure tag by default and sets an alternate text of the figure from the $\langle text \rangle$.

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

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

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

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

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

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

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

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

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

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

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

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

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

You can set these options anywhere in the document by declaring `\SetKeys[luamplib/tagging]{<key-val list>}`, which will affect `mplib` figures thereafter in the scope. And the options listed above are provided for `\mpfig` and `\usemplibgroup` (see [below § 1.2.13](#)) commands as well.

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

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

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

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

```

\end{mplibcode}

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

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

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

```

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

1.2 METAPOST

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

`mplibdimen <string>` and `mplibcolor <string>` 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 <string>` 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.⁹ For instance:

```

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

```

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

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

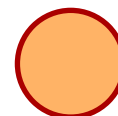
1.2.3 `withmplibcolors (...)`

Unlike the `withcolor` command, users can specify one color for filling and another color for stroking using the macro `withmplibcolors` at the end of a sentence. The syntax is `withmplibcolors`

⁹Since 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.

(*fill color expr*), (*stroke color expr*)). When the argument is in string type, it is regarded as the color expression of \TeX side. A simple example (see also the example at § 1.2.10):

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

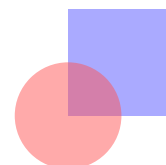


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 withtransparency (... , ...)

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

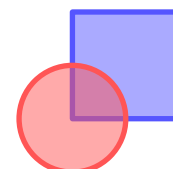
```
\mpfig
fill unitsquare scaled 40
  withcolor 1/3[blue,white]
  withtransparency (1, 0.5)      % or ("normal", 0.5)
;
fill fullcircle scaled 40
  withcolor 1/3[red,white]
  withtransparency (1, 0.5)
;
\endmpfig
```



1.2.5 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 ... 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 § 1.2.8), 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 (e.g., see below § 1.2.10, particularly the first *example* of tiling pattern at § 1.2.12; see also § 1.2.13 and § 1.2.14).

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



- When shading a picture generated by 'infont' operator or that has multiple components, the effect of *withshadingvector* and that of *withshadingdirection* will be the same, as *luamplib* considers only the bounding box of the picture.
- A few more optional macros are available in addition the *metafun*'s: *withshadingpoints*, *withshadingcenters*, *withshadingextend*, *withshadingstroke*, and *withshadingmatrix*.

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

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

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

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

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

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

withshadingfactor $\langle numeric \rangle$ Multiplier of the radii, 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 declared.

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.

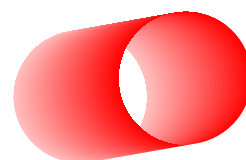
withshadingfraction $\langle numeric \rangle$ Fractional number of each shading step, and so only meaningful within withshadingstep.

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

withshadingdomain $\langle pair \rangle$ 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". Only meaningful when the object is a $\langle \text{picture} \rangle$.

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.

1.2.7 ... 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 § 1.2.15.

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 $\langle \text{string} \rangle$ 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 ($\langle \text{pair} \rangle$, $\langle \text{pair} \rangle$) sets the bounding box of the fading area, default value being (llcorner p, urcorner p). Though this option is not needed in most cases, there could be cases when users want to explicitly control the bounding box. Particularly, see the description [below](#) at § 1.2.13 on the analogous macro withgroupbbox.

An example:

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



1.2.8 mlibgraphicx ...

`mlibgraphicx` $\langle string \rangle$ is a METAPOST operator, the effect of which is similar to that of ConTeXt's `graphicx` or our own `mliboutlinetext` (see below § 1.2.11). However the syntax is somewhat different.

```
draw mlibgraphicx "$\sqrt{2+x}$"
  rotated 10 scaled 3
  fakebold 2.5
  fillcolor "red!50"
  drawcolor 2/3 red
  ;
```



`fakebold`, `fillcolor` and `drawcolor` (or `strokecolor`) are optional; default values are 2, "white" and "black" respectively.¹⁰ When the color expression is given in string type, it is regarded as color, `xcolor` or `l3color`'s expression. All from `mlibgraphicx` 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 `graphicx`.

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

1.2.9 mlibglyph ... of ...

METAPOST operator `mlibglyph` $\langle number \rangle$ | $\langle string \rangle$ 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.

```
mlibglyph 50 of \fontid\font % slot 50 of current font
mlibglyph "Q" of "TU/TeXGyrePagella(0)/m/n/10" % font csname
mlibglyph "Q" of "texgyrepagella-regular.otf" % raw filename
mlibglyph "R" of "utmr8a.pfb" % raw filename (type1 font)
```

¹⁰Users can use the `withmlibcolors` macro instead of `fillcolor` and `drawcolor` options. See § 1.2.3 on this macro.

¹¹But this limitation is now lifted by the introduction of `withshadingmethod`. See above § 1.2.6.


```

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.¹² If that happens, consider using glyph operator instead of `mplibglyph`.

1.2.10 `mplibdrawglyph ...`, `mplibstrokeglyph ...`, `mplibfillandstrokeglyph ...`

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

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

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

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

An example:

```

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

```



1.2.11 `mpliboutlinetext (...)`

As said before at § 1.1.3, `luamplib` provides the METAPOST operator `mpliboutlinetext` (*<string>*) which mimicks *metafun*’s `outlinetext`, but with some enhancements including the support for

¹²The 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 \LaTeX format.

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

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.

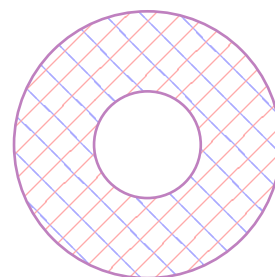
1.2.12 `\mppattern{...} ... \endmppattern, ... withmppattern ...`

TeX macros `\mppattern{<name>} ... \endmppattern` define a tiling pattern cell associated with the `<name>`. METAPOST command `withmppattern`, the syntax being `<cyclic path> | <textual picture> withmppattern <string>`, will fill the given path or text with the tiling pattern cell of the `<name>` by replicating it horizontally and vertically.¹⁴ As said before at § 1.2.6, the *textual picture* here means any text typeset by TeX, mostly the result of the `btex` command (and its derivatives) or the `infont` operator.

An example:

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

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



¹⁴`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, `<cyclic path> withmppattern <string>` works as intended only with `fill` or `filldraw` command.

Table 1: options for \mppattern

| Key | Value Type | Explanation |
|---------------------|------------------------|---|
| xstep | <i>number</i> | horizontal spacing between pattern cells |
| ystep | <i>number</i> | vertical spacing between pattern cells |
| xshift | <i>number</i> | horizontal shifting of pattern cells |
| yshift | <i>number</i> | vertical shifting of pattern cells |
| bbox | <i>table 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

```

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).¹⁵ 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");
for i=1 upto mpliboutlinenum:

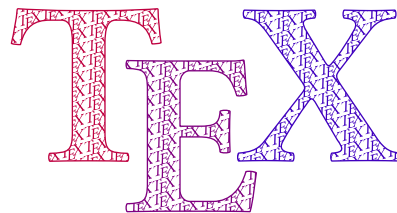
```

¹⁵When using DVI mode, `-c` option might be needed to the `dvipdfmx` command.

```

mplibfillandstrokeglyph mpliboutlinepic[i]
scaled 8
withmppattern "pattnocolor"
withpen pencircle scaled 1/2
withcolor (i/4)[red,blue] % paints the pattern
;
endfor
endfig;
\end{mplibcode}

```



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

```

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

```



1.2.13 ... 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 `""` | `"isolated"` | `"knockout"` | `"isolated,knockout"` | `"off"`, which will return a METAPOST picture. The additional features provided by *luamplib* are:

- As shown, in addition to those arguments mimicking *metafun*'s, we allow another argument at the right-hand side: `asgroup "off"` will produce an ordinary *form XObject* rather than a transparency group XObject. On the contrary, `asgroup ""` (empty string) will produce a transparency group in which both of the PDF keys `/I` and `/K` are false.
- 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 `<string>` 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{⟨name⟩}` 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 ⟨string⟩` 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.

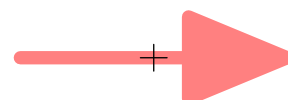
`withgroupbbox (⟨pair⟩, ⟨pair⟩)` sets the bounding box of the XObject, default value being (llcorner p, urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence ‘withgroupbbox (bot lft llcorner 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:

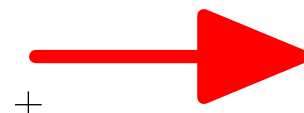
```
\mpfig
picture pic;
pic = image(drawarrow (left--right) scaled 5 withcolor red) scaled 10 ;
draw pic
  asgroup "off"
  withtransparency (1, 1/2) ;
\endmpfig
```



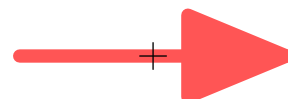
```
\mpfig
draw pic
  asgroup ""
  withgroupname "mygroup"
  withtransparency (1, 1/2) ;
draw (left--right) scaled 5 ;
draw (up--down) scaled 5 ;
\endmpfig
```



```
\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
```

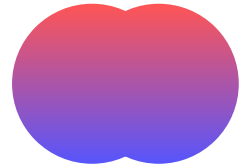


Also note that normally the XObjects are not affected by outer color commands. However, if you have made the original XObject using `withoutcolor` command, colors will have effects on its uncolored objects.

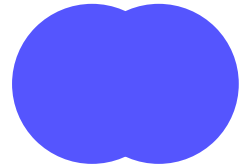
N.B. When you give shading effect upon a *textual picture* (i.e. non-path object) inside or outside a transparency group, currently many of the PDF renderers, including Mac OS Preview and Foxit Editor, do not interpret PDF coordinates properly. If that happens, consider using other PDF viewer such as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

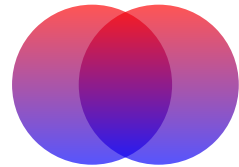


```
\mpfig                                % shading outside group
draw pic1
asgroup ""
withshadingmethod "linear"
withshadingvector (2, 1)
withshadingcolors (red, blue)
withtransparency (1, 2/3);
\endmpfig
```



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 in a transparency group XObject. Most PDF renderers do render it properly:

```
\mpfig                                % shading within ordinary xobject
draw pic2
asgroup "off"
withgroupname "myshading"
withtransparency (1, 2/3);
\endmpfig
```



```
\mpfig                                % wrap in a transparency group
usemplibgroup "myshading"
asgroup ""
withtransparency (1, 2/3);
\endmpfig
```



or preferably (see next subsection § 1.2.14 on `\mplibgroup`):

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

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



1.2.14 `\mplibgroup{...} ... \endmplibgroup`

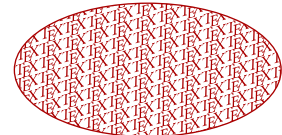
These T_EX macros are described here in this subsection, as they are deeply related to the `asgroup` operator described just above at § 1.2.13. Users can define a transparency group or an ordinary *form XObject* with these macros from T_EX side. The syntax is similar to the `\mppattern` command (see above § 1.2.12).

An example:¹⁶

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

\usemplibgroup{texpatt}

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



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

When `asgroup` option is not given or is given as "off", an ordinary *form XObject* will be generated rather than a transparency group. Thus the individual objects, not the *XObject* as a

¹⁶Note that, by the option `asgroup="wrapped"`, within a transparency group *XObject* a tiling pattern is wrapped 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" |
| 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

whole, will be affected by outer transparency command, just like the first figure in the [example](#) above at § 1.2.13.

Option asgroup="wrapped" can be regarded as a shortcut of double \mplibgroup command. The content will be wrapped in an ordinary formXObject before being wrapped again in a transparency group. This option could be useful when a tiling pattern (see the example just above) or a picture shading (see the [example](#) above at § 1.2.13) is used in the content.

As for the option asgroup="masking", see the next subsection § 1.2.15.

With colorspace option, which is no-op for ordinary formXObject, users can specify the color space of a transparency group. For instance, the mplibgroup will be painted in grayscale model when colorspace="/DeviceGray" is given to an *isolated* transparency group.¹⁷

As shown, you can reuse the mplibgroup using the T_EX command \usemplibgroup or the METAPOST command usemplibgroup. The behavior of these commands is the same as that described [above](#) at § 1.2.13, excepting that the mplibgroup made by T_EX code (not by METAPOST code) respects original height and depth.

1.2.15 ... withmaskinggroup ...

Using this command, the mplibgroup (see above § 1.2.14) generated by the option asgroup="masking" (see Table 2) can be utilized as a masking transparency group upon a picture or a path object. The syntax is $\langle picture \rangle | \langle path \rangle$ withmaskinggroup $\langle string \rangle$, the latter being the name of a pre-defined masking group.

Basically, the masking group should be prepared in *grayscale* color model: the area painted with 1 (\approx white: full luminosity) will preserve the full color of the object; the area painted with 0 (\approx black: zero luminosity) will force full transparency, masking it invisibly.¹⁸

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.¹⁹

¹⁷Note that some PDF renderers such as Mac OS Preview or Firefox do not render properly this option.

¹⁸In 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 § 1.2.14).

¹⁹Color expressions in rgb or cmyk model are also allowed, in accordance with the option colorspace="/DeviceRGB"

An example:

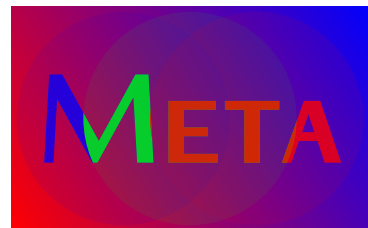
```

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

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

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

```



1.2.16 `mpliblength ...`, `mplibuclength ...`

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

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

1.2.17 `mplibsubstring ... of ...`, `mplibucsubstring ... of ...`

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

or `colorspace="/DeviceCMYK"` given to the masking transparency group. This however we do not recommend as the luminosity is difficult to understand intuitively.

On the other hand, `mplibucsubstr` $\langle pair \rangle$ of $\langle string \rangle$ returns the part of the string indexed by unicode grapheme clusters. For instance, `mplibucsubstr (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.

1.3 Lua

1.3.1 runscript ...

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

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

1.3.2 Lua table `luamplib.instances`

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

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

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

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

```
\def\mpnumeric#1#2{\directlua{
```

Table 3: elements in luamplib table (partial)

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

```

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 § 1.2.12.

```
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 0.²⁰

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

²⁰In DVI mode, T_EX macro ‘`mplibpatternname`’ should be set as $\langle pattern\ name \rangle$ before preparing the box, if shading pattern (i.e. shading on picture) is used in the pattern cell.

1.3.5 Lua function `luamplib.registergroup`

This is the Lua interface for `\mplibgroup ... \endmplibgroup` described above at § 1.2.14.

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

The first argument is the register of a box prepared in advance by the user. When the contents of the box have been generated from \TeX (not \METAPOST) code, please make sure that both of the \TeX macros ‘`MPllx`’ and ‘`MPlly`’ are defined as ‘`0`’ before invoking the Lua function.²¹

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   = "2.41.2",
5   date      = "2026/05/12",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8
```

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

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

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+"

```

²¹In 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`.

```

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

```

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

```

71 local file = file or { }
72 local replacesuffix = file.replacesuffix or function(filename, suffix)
73   return (filename:gsub("%.[%a%d]+$", "")) .. "." .. suffix
74 end
75 local is_writable = file.is_writable or function(name)
76   if lfs.isdir(name) then
77     name = name .. "_luamplib_temp_file_"
78     local fh = io.open(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.mkdir or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\w/]+)") do
88     full = full .. sub
89     lfs.mkdir(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 lfs.isdir(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 verbatimtex_etex = name_b.."verbatimtex"..name_e.."s*(.)%s*"..name_b.."etex"..name_e
151

```

Function luamplib.finder

```

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

```

format.mp is much complicated, so specially treated.

```

155 local function replaceformatmp(file,newfile,ofmodify)
156   local fh = ioopen(file,"r")
157   if not fh then return file end
158   local data = fh:read("*all"); fh:close()

```

```

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

```

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

```

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

```



```

205     lfstouch(newfile,currenttime,ofmodify)
206     return newfile
207 end

```

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

```

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

```

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

```

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

```

Make use of `make_text` and `run_script`. 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.legacyverbatimtex and luamplib.preambles.legacyverbatimtex 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                 luamplib.numbersystem or "scaled",
325                 tostring(luamplib.texttextlabel),
326                 tostring(luamplib.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 luamplib.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 = luamplibload(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 = luamplib.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_T_EX'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"^%d" 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@color\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   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('1 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 \TeX 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 ConTeXt 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.legacyverbatimtex = 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("%zPerCent%z", "\\%\%")

```

```

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.legacyverbatim then
611                         if luamplib.in_the_fig then
612                             return process_verbatimtex_infig(str)
613                         else
614                             return process_verbatimtex_prefig(str)
615                         end
616                     else
617                         return process_verbatimtex_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:nn{]], 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 graphictextcolor (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 = graphictextcolor(fill, "fill")
698     stroke = graphictextcolor(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 mplibgraphictext 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 mplibgraphictext

```

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

    local k = format("shapes_%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

```

mpliboutlinetext : 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

```

```

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 = luampplib.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 graphictextpic: 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 mplibgraphictext 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 luampplib.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   else :
1681     withprescript "sh_type=circular"
1682     withprescript "sh_factor=1.2"
1683     withprescript "sh_center_a=" & ddecimal center p
1684     withprescript "sh_center_b=" & ddecimal center p
1685     withprescript "sh_radius_a=" & decimal 0
1686     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1687   fi
1688 enddef;
1689 def mplib_with_shade_method_analyze(expr p) =
1690   mplib_shade_path := p ;
1691   mplib_shade_step := 1 ;
1692   mplib_shade_fx := xpart point 0 of p ;
1693   mplib_shade_fy := ypart point 0 of p ;
1694   mplib_shade_lx := mplib_shade_fx ;
1695   mplib_shade_ly := mplib_shade_fy ;
1696   mplib_shade_nx := 0 ;
1697   mplib_shade_ny := 0 ;
1698   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1699   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1700   for i=1 upto length(p) :
1701     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1702     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1703     if mplib_shade_tx > mplib_shade_dx :
1704       mplib_shade_nx := i + 1 ;
1705       mplib_shade_lx := xpart point i of p ;
1706       mplib_shade_dx := mplib_shade_tx ;
1707     fi ;
1708     if mplib_shade_ty > mplib_shade_dy :
1709       mplib_shade_ny := i + 1 ;
1710       mplib_shade_ly := ypart point i of p ;
1711       mplib_shade_dy := mplib_shade_ty ;
1712     fi ;
1713   endfor ;
1714 enddef;
1715 vardef mplib_max_radius(expr p) =
1716   max (
1717     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1718     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1719     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1720     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1721   )
1722 enddef;
1723 def withshadingstep (text t) =
1724   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1725   withprescript "sh_step=" & decimal mplib_shade_step
1726   t

```

```

1727 enddef;
1728 let withfadestep = withshadingstep;
1729 def withshadingradius expr a =
1730   withprescript "sh_radius_a=" & decimal (xpart a)
1731   withprescript "sh_radius_b=" & decimal (ypart a)
1732 enddef;
1733 def withshadingorigin expr a =
1734   withprescript "sh_center_a=" & ddecimal a
1735   withprescript "sh_center_b=" & ddecimal a
1736 enddef;
1737 def withshadingvector expr a =
1738   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1739   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1740 enddef;
1741 def withshadingdirection expr a =
1742   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1743   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1744 enddef;
1745 def withshadingtransform expr a =
1746   withprescript "sh_transform=" & a
1747 enddef;
1748 def withshadingcenter expr a =
1749   withprescript "sh_center_a=" & ddecimal (
1750     center mplib_shade_path shifted (
1751       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1752       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1753     )
1754   )
1755 enddef;
1756 def withshadingcenters (expr a, b) =
1757   withprescript "sh_center_a=" & ddecimal a
1758   withprescript "sh_center_b=" & ddecimal b
1759   withshadingtransform "no"
1760   withshadingfactor 1
1761 enddef;
1762 let withshadingpoints = withshadingcenters;
1763 def withshadingextend (expr a, b) =
1764   withprescript "sh_extend=" &
1765     if a: "true" else: "false" fi & " " &
1766     if b: "true" else: "false" fi
1767 enddef;
1768 let withfadeextend = withshadingextend;
1769 def withshadingdomain expr d =
1770   withprescript "sh_domain=" & ddecimal d
1771 enddef;
1772 def withshadingfactor expr f =
1773   withprescript "sh_factor=" & decimal f
1774 enddef;
1775 def withshadingfraction expr a =

```

```

1776 if mplib_shade_step > 0 :
1777     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1778 fi
1779 enddef;
1780 let withfadefraction = withshadingfraction;
1781 def withshadingcolors (expr a, b) =
1782     if mplib_shade_step > 0 :
1783         withprescript "sh_color=into"
1784         withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1785         withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1786     else :
1787         withprescript "sh_color=into"
1788         withprescript "sh_color_a=" & colordecimals a
1789         withprescript "sh_color_b=" & colordecimals b
1790     fi
1791 enddef;
1792 let withfadeopacity = withshadingcolors;
1793 def withshadingstroke expr a =
1794     withprescript "sh_stroking=" & a
1795 enddef;
1796 def withshadingmatrix expr s =
1797     withprescript "sh_matrix=" & s
1798 enddef;
1799 let withfadematrix = withshadingmatrix;
1800 def mpliblength primary t =
1801     runscript("return utf8.len[==[" & t & "]==]")
1802 enddef;
1803 def mplibsubstring expr p of t =
1804     runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1805         & decimal xpart p & ",",
1806         & decimal ypart p & ")")
1807 enddef;
1808 def mplibuclength primary t =
1809     runscript("return #luamplib.getunicodegraphemes[==[" & t & "]==]")
1810 enddef;
1811 def mplibucsubstring expr p of t =
1812     runscript("return luamplib.unicodesubstring([==[" & t & "]==],",
1813         & decimal xpart p & ",",
1814         & decimal ypart p & ",true)")
1815 enddef;
1816 ]],
1817 legacyverbatimtex = [[
1818 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&"}") enddef;
1819 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&"}") enddef;
1820 let VerbatimTeX = specialVerbatimTeX;
1821 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1822     "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ")";
1823 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1824     "runscript(" &ditto&

```

```

1825 "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1826 "luamplib.in_the_fig=false" &ditto& ");";
1827 ]],
1828 texttextlabel = [[
1829 let luampliboriginalinfont = infont;
1830 primarydef s infont f =
1831   if (s < char 32)
1832     or (s = char 35) % #
1833     or (s = char 36) % $
1834     or (s = char 37) % %
1835     or (s = char 38) % &
1836     or (s = char 92) % \
1837     or (s = char 94) % ^
1838     or (s = char 95) % _
1839     or (s = char 123) % {
1840     or (s = char 125) % }
1841     or (s = char 126) % ~
1842     or (s = char 127) :
1843     s luampliboriginalinfont f
1844   else :
1845     rawtexttext(s)
1846   fi
1847 enddef;
1848 def fontsize expr f =
1849   begingroup
1850   save size; numeric size;
1851   size := mplibdimen("1em");
1852   if size = 0: 10pt else: size fi
1853   endgroup
1854 enddef;
1855 ]],
1856 }
1857

```

process_mplibcode

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

```

1858 luamplib.verbatiminput = false
1859 luamplib.everymplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1860 luamplib.everyendmplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1861 function luamplib.process_mplibcode (data, instancename)
1862   texboxes.localid = 4096

```

This is needed for legacy behavior

```

1863 if luamplib.legacyverbatimtex then
1864   luamplib.figid, tex_code_pre_mplib = 1, {}
1865 end
1866 local everymplib = luamplib.everymplib[instancename]
1867 local everyendmplib = luamplib.everyendmplib[instancename]
1868 data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
1869 :gsub("\r","\n")

```

These five lines are needed for mplibverbatim mode.

```

1870 if luamplib.verbatiminput then
1871   data = data:gsub("\\mpcolor%s+(-%b{})", "mplibcolor(\"%1\")")
1872   :gsub("\\mpdim%s+(%b{})", "mplibdimen(\"%1\")")
1873   :gsub("\\mpdim%s+(\\%a+)", "mplibdimen(\"%1\")")
1874   :gsub(btex_etex, "btex %1 etex ")
1875   :gsub(verbatimtex_etex, "verbatimtex %1 etex;")
1876 else

```

If not mplibverbatim, expand mplibcode data, so that users can use T_EX 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.

```

1877   local t = { } -- to store btex, verbatimtex, string
1878   data = data:gsub(btex_etex, function(str)
1879     t[#t+1] = str
1880     return format("btex \\unexpanded{!!u!a!s!m!p!!} etex ", #t) -- space
1881   end)
1882   :gsub(verbatimtex_etex, function(str)
1883     t[#t+1] = str
1884     return format("verbatimtex \\unexpanded{!!u!a!s!m!p!!} etex;", #t) -- semicolon
1885   end)
1886   :gsub('\"(.-)\"', function(str)
1887     t[#t+1] = str
1888     return format('\"\\unexpanded{!!u!a!s!m!p!!}\"', #t)
1889   end)
1890   :gsub("\\%", "\\0PerCent\\0")
1891   :gsub("%%.-\\n", "\\n")
1892   :gsub("%zPerCentz", "\\%")
1893   run_tex_code(format("\\mplibtmptoks\\expandafter{\\expanded{%s}}", data))
1894   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

1895   :gsub("##", "#")
1896   :gsub("!!u!a!(%d+)!m!p!!", function(str) return t[tonumber(str)] or str end)
1897 end
1898 process(data, instancename)
1899 end
1900

```

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.

```

1901 local figcontents = { post = { } }
1902 local function put2output(a,...)
1903   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
1904 end
1905 local function pdf_startfigure(n,llx,lly,urx,ury)
1906   put2output("\\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
1907 end
1908 local function pdf_stopfigure()
1909   put2output("\\mplibstoptoPDF")

```

```
1910 end
```

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

```
1911 local function pdf_literalcode (...)
1912   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
1913 end
1914 local start_pdf_code = pdfmode
1915   and function() pdf_literalcode"q" end
1916   or function() put2output"\special{pdf:bcontent}" end
1917 local stop_pdf_code = pdfmode
1918   and function() pdf_literalcode"Q" end
1919   or function() put2output"\special{pdf:econtent}" end
1920
```

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

```
1921 local function put_tex_boxes (object,prescript)
1922   local box = prescript.mplibtexboxid:explode":"
1923   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
1924   if n and tw and th then
1925     local op = object.path
1926     local first, second, fourth = op[1], op[2], op[4]
1927     local tx, ty = first.x_coord, first.y_coord
1928     local sx, rx, ry, sy = 1, 0, 0, 1
1929     if tw ~= 0 then
1930       sx = (second.x_coord - tx)/tw
1931       rx = (second.y_coord - ty)/tw
1932       if sx == 0 then sx = 0.00001 end
1933     end
1934     if th ~= 0 then
1935       sy = (fourth.y_coord - ty)/th
1936       ry = (fourth.x_coord - tx)/th
1937       if sy == 0 then sy = 0.00001 end
1938     end
1939
```

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

```
1940   local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
1941   put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
1942 end
1943 end
1944
```

Colors

```
1945 local do_preobj_CR
1946 do
1947   local prev_override_color
1948   function do_preobj_CR(object,prescript)
1949     if object.postscript == "collect" then return end
1950     local override = prescript and prescript.mpliboverridecolor
1951     if override then
1952       if pdfmode then
```

```

1953     pdf_literalcode(override)
1954     override = nil
1955   else
1956     put2output("\\special{%s}",override)
1957     prev_override_color = override
1958   end
1959 else
1960   local cs = object.color
1961   if cs and #cs > 0 then
1962     pdf_literalcode(luamplib.colorconverter(cs))
1963     prev_override_color = nil
1964   elseif not pdfmode then
1965     override = prev_override_color
1966     if override then
1967       put2output("\\special{%s}",override)
1968     end
1969   end
1970 end
1971 return override
1972 end
1973 end
1974

```

For transparency, shading, fading, and pattern

```

1975 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
1976 local pdfobjs, pdfetcs = {}, {}
1977 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"
1978 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
1979 pdfetcs.pgfcsp = "pgf@sys@addpdfresource@colorspaces@plain"
1980 local function update_pdfobjs (os, stream)
1981   local key = os
1982   if stream then key = key..stream end
1983   local on = key and pdfobjs[key]
1984   if on then
1985     return on,false
1986   end
1987   if pdfmode then
1988     if stream then
1989       on = pdf.immediateobj("stream",stream,os)
1990     elseif os then
1991       on = pdf.immediateobj(os)
1992     else
1993       on = pdf.reserveobj()
1994     end
1995   else
1996     on = pdfetcs.cnt or 1
1997     if stream then
1998       texsprint(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
1999     elseif os then

```



```

2000     texsprint(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2001   else
2002     texsprint(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2003   end
2004   pdfetcs.cnt = on + 1
2005 end
2006 if key then
2007   pdfobjs[key] = on
2008 end
2009 return on,true
2010 end
2011 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2012 if pdfmode then
2013   pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2014   local getpagers = pdfetcs.getpagers
2015   local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2016   local initialize_resources = function (name)
2017     local tabname = format("%s_res",name)
2018     pdfetcs[tabname] = { }
2019     if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
2020       local obj = pdf.reserveobj()
2021       setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
2022       luatexbase.add_to_callback("finish_pdffile", function()
2023         pdf.immediateobj(obj, format("<<s>>", tableconcat(pdfetcs[tabname])))
2024       end,
2025       format("luamplib.%s.finish_pdffile",name))
2026     end
2027   end
2028   pdfetcs.fallback_update_resources = function (name, res)
2029     local tabname = format("%s_res",name)
2030     if not pdfetcs[tabname] then
2031       initialize_resources(name)
2032     end
2033     if luatexbase.callbacktypes.finish_pdffile then
2034       local t = pdfetcs[tabname]
2035       t[#t+1] = res
2036     else
2037       local tpr, n = getpagers() or "", 0
2038       tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2039       if n == 0 then
2040         tpr = format("%s/%s<<s>>", tpr, name, res)
2041       end
2042       setpagers(tpr)
2043     end
2044   end
2045 else
2046   texsprint {
2047     "\\luamplibatfirstshipout{",
2048     "\\special{pdf:obj @MPLibTr<<>>}",

```

```

2049     "\\special{pdf:obj @MPLibSh<<>>}",
2050     "\\special{pdf:obj @MPLibCS<<>>}",
2051     "\\special{pdf:obj @MPLibPt<<>>}}",
2052 }
2053 pdfetcs.fallback_update_resources = function (name,res,obj)
2054     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2055     local tabname = format("%s_res",name)
2056     if not pdfetcs[tabname] then
2057         texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2058         pdfetcs[tabname] = { }
2059     end
2060     tableinsert(pdfetcs[tabname], res)
2061 end
2062 end
2063

```

Transparency

```

2064 local function add_extgs_resources (on, new)
2065     local key = format("MPLibTr%s", on)
2066     if new then
2067         local val = format(pdfetcs.resfmt, on)
2068         if pdfmanagement then
2069             texsprint {
2070                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2071             }
2072         else
2073             local tr = format("/%s %s", key, val)
2074             if is_defined(pdfetcs.pgfgextgs) then
2075                 texsprint { "\\csname ", pdfetcs.pgfgextgs, "\\endcsname{" , tr, "}" }
2076             elseif is_defined"TRP@list" then
2077                 texsprint(catat11,{
2078                     [[\if@files\immediate\write\@auxout{]],
2079                     [[\string\g@addto@macro\string\TRP@list{]],
2080                     tr,
2081                     [[}]\fi]],
2082                 })
2083                 if not get_macro"TRP@list":find(tr) then
2084                     texsprint(catat11,[[\global\TRP@reruntrue]])
2085                 end
2086             else
2087                 pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2088             end
2089         end
2090     end
2091     return key
2092 end
2093
2094 local do_preobj_TR
2095 do

```

```

2096 local transparency_modes = {
2097   [0] = "Normal",
2098   "Normal",      "Multiply",      "Screen",      "Overlay",
2099   "SoftLight",    "HardLight",    "ColorDodge",  "ColorBurn",
2100   "Darken",       "Lighten",       "Difference",  "Exclusion",
2101   "Hue",          "Saturation",    "Color",      "Luminosity",
2102   "Compatible",
2103   normal      = "Normal",    multiply = "Multiply",    screen = "Screen",
2104   overlay     = "Overlay",    softlight = "SoftLight",  hardlight = "HardLight",
2105   colordodge  = "ColorDodge", colorburn = "ColorBurn",  darken = "Darken",
2106   lighten     = "Lighten",    difference = "Difference", exclusion = "Exclusion",
2107   hue         = "Hue",        saturation = "Saturation", color = "Color",
2108   luminosity  = "Luminosity", compatible = "Compatible",
2109 }
2110 function do_preobj_TR(object,prescript)
2111   if object.postscript == "collect" then return end
2112   local opaq = prescript and prescript.tr_transparency
2113   if not opaq then return end
2114
2115   local key, on, os, new
2116   local mode = prescript.tr_alternative or 1
2117   mode = transparency_modes[tonumber(mode) or mode:lower()]
2118   if not mode then
2119     mode = prescript.tr_alternative
2120     warn("unsupported blend mode: '%s'", mode)
2121   end
2122   opaq = opaq:explode":""
2123   for i,v in ipairs(opaq) do
2124     opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2125   end
2126   for i,v in ipairs{ {mode,opaq[1],opaq[2] or opaq[1]},{"Normal",1,1} } do
2127     os = format("</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2128     on, new = update_pdfobjs(os)
2129     key = add_extgs_resources(on,new)
2130     if i == 1 then
2131       pdf_literalcode("/%s gs",key)
2132     else
2133       return format("/%s gs",key)
2134     end
2135   end
2136 end
2137 end
2138

```

Shading with *metafun* format.

```

2139 local function sh_pdfpageresources(shtype, domain, colorspace, ca, cb, coordinates, steps, fractions, extend)
2140   for _,v in ipairs{ca,cb} do
2141     for i,vv in ipairs(v) do
2142       for ii,vvv in ipairs(vv) do

```

```

2143     v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2144     end
2145   end
2146 end
2147 local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
2148 if steps > 1 then
2149   local list,bounds,encode = { },{ },{ }
2150   for i=1,steps do
2151     if i < steps then
2152       bounds[i] = format("%.3f", fractions[i] or 1)
2153     end
2154     encode[2*i-1] = 0
2155     encode[2*i]   = 1
2156     os = fun2fmt:format(domain,tableconcat(ca[i], ' '),tableconcat(cb[i], ' '))
2157       :gsub(decimals,rmzeros)
2158     list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2159   end
2160   os = tableconcat {
2161     "<</FunctionType 3",
2162     format("/Bounds[%s]",   tableconcat(bounds, ' ')),
2163     format("/Encode[%s]",   tableconcat(encode, ' ')),
2164     format("/Functions[%s]", tableconcat(list, ' ')),
2165     format("/Domain[%s]>>", domain),
2166   } :gsub(decimals,rmzeros)
2167 else
2168   os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
2169   :gsub(decimals,rmzeros)
2170 end
2171 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2172 os = tableconcat {
2173   format("<</ShadingType %i", shtype),
2174   format("/ColorSpace %s",   colorspace),
2175   format("/Function %s",     objref),
2176   format("/Coords[%s]",     coordinates),
2177   format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2178 } :gsub(decimals,rmzeros)
2179 local on, new = update_pdfobjs(os)
2180 if new then
2181   local key, val = format("MPlibSh%s", on), format(pdfetcs.resfmt, on)
2182   if pdfmanagement then
2183     texsprint {
2184       "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2185     }
2186   else
2187     local res = format("/%s %s", key, val)
2188     pdfetcs.fallback_update_resources("Shading",res,"@MPlibSh")
2189   end
2190 end
2191 return on

```

```

2192 end
2193
2194 local do_preobj_SH
2195 do
2196   pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
2197     run_tex_code({
2198       [[\color_model_new:nnn]],
2199       format("{mplibcolorspace_%s}", names:gsub(",","_")),
2200       format("{DeviceN}{names={%s}}", names),
2201       [[\edef\mplib@tempa{\pdf_object_ref_last:}]],
2202     }, ccexplat)
2203     local colorspace = get_macro'mplib@tempa'
2204     t[names] = colorspace
2205     return colorspace
2206   end })
2207   local function color_normalize(ca,cb)
2208     if #cb == 1 then
2209       if #ca == 4 then
2210         cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2211       else -- #ca = 3
2212         cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2213       end
2214     elseif #cb == 3 then -- #ca == 4
2215       cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
2216     end
2217   end
2218   function do_preobj_SH(object, prescript)
2219     local shade_no
2220     local sh_type = prescript and prescript.sh_type
2221     if not sh_type then return end
2222
2223     local domain = prescript.sh_domain or "0 1"
2224     local centera = (prescript.sh_center_a or "0 0"):explode()
2225     local centerb = (prescript.sh_center_b or "0 0"):explode()
2226     local transform = prescript.sh_transform == "yes"
2227     local sx,sy,sr,dx,dy = 1,1,1,0,0
2228     if transform then
2229       local first = (prescript.sh_first or "0 0"):explode()
2230       local setx = (prescript.sh_set_x or "0 0"):explode()
2231       local sety = (prescript.sh_set_y or "0 0"):explode()
2232       local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2233       if x ~= 0 and y ~= 0 then
2234         local path = object.path
2235         local path1x = path[1].x_coord
2236         local path1y = path[1].y_coord
2237         local path2x = path[x].x_coord
2238         local path2y = path[y].y_coord
2239         local dxa = path2x - path1x
2240         local dya = path2y - path1y

```

```

2241     local dxb = setx[2] - first[1]
2242     local dyb = sety[2] - first[2]
2243     if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2244         sx = dxa / dxb ; if sx < 0 then sx = - sx end
2245         sy = dya / dyb ; if sy < 0 then sy = - sy end
2246         sr = math.sqrt(sx^2 + sy^2)
2247         dx = path1x - sx*first[1]
2248         dy = path1y - sy*first[2]
2249     end
2250 end
2251 end
2252 local ca, cb, colorspace, steps, fractions
2253 ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode":" }
2254 cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode":" }
2255 steps = tonumber(prescript.sh_step) or 1
2256 if steps > 1 then
2257     fractions = { prescript.sh_fraction_1 or 0 }
2258     for i=2,steps do
2259         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2260         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode":"
2261         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode":"
2262     end
2263 end
2264 if prescript.mpllib_spotcolor then
2265     ca, cb = { }, { }
2266     local names, pos, objref = { }, -1, ""
2267     local script = object.prescript:explode"\13+"
2268     for i=#script,1,-1 do
2269         if script[i]:find"mpllib_spotcolor" then
2270             local t, name, value = script[i]:explode"="[2]:explode":"
2271             value, objref, name = t[1], t[2], t[3]
2272             if not names[name] then
2273                 pos = pos+1
2274                 names[name] = pos
2275                 names[#names+1] = name
2276             end
2277             t = { }
2278             for j=1,names[name] do t[#t+1] = 0 end
2279             t[#t+1] = value
2280             tableinsert(#ca == #cb and ca or cb, t)
2281         end
2282     end
2283     for _,t in ipairs{ca,cb} do
2284         for _,tt in ipairs(t) do
2285             for i=1,#names-#tt do tt[#tt+1] = 0 end
2286         end
2287     end
2288     if #names == 1 then
2289         colorspace = objref

```

```

2290     else
2291         colorspace = pdfetcs.clrspcs[ tableconcat(names,"") ]
2292     end
2293 else
2294     local model = 0
2295     for _,t in ipairs{ca,cb} do
2296         for _,tt in ipairs(t) do
2297             model = model > #tt and model or #tt
2298         end
2299     end
2300     for _,t in ipairs{ca,cb} do
2301         for _,tt in ipairs(t) do
2302             if #tt < model then
2303                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2304             end
2305         end
2306     end
2307     colorspace = model == 4 and "/DeviceCMYK"
2308                 or model == 3 and "/DeviceRGB"
2309                 or model == 1 and "/DeviceGray"
2310                 or err"unknown color model"
2311 end
2312 local extend = prescript.sh_extend
2313 if sh_type == "linear" then
2314     local coordinates = format("%f %f %f %f",
2315         dx + sx*centera[1], dy + sy*centera[2],
2316         dx + sx*centerb[1], dy + sy*centerb[2])
2317     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2318 elseif sh_type == "circular" then
2319     local factor = prescript.sh_factor or 1
2320     local radiusa = factor * prescript.sh_radius_a
2321     local radiusb = factor * prescript.sh_radius_b
2322     local coordinates = format("%f %f %f %f %f %f",
2323         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2324         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2325     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2326 else
2327     err"unknown shading type"
2328 end
2329 return shade_no, prescript.sh_stroking == "yes"
2330 end
2331 end
2332

```

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

```

2333 local function add_pattern_resources (key, val)
2334 if pdfmanagement then
2335     textsprint {
2336         "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{", val, "}"

```

```

2337 }
2338 else
2339   local res = format("/%s %s", key, val)
2340   if is_defined(pdfetcs.pgfpattern) then
2341     texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{", res, "}" }
2342   else
2343     pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2344   end
2345 end
2346 end
2347 if pdfmode then
2348   function luamplib.dolatelua (on, os, matrix)
2349     local h, v = pdf.getpos()
2350     h = format("%f", h/factor) :gsub(decimals,rmzeros)
2351     v = format("%f", v/factor) :gsub(decimals,rmzeros)
2352     local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2353     matrix = format("%s %s %s %s %s %s", t[1], t[2], t[3], t[4], t[5]+h, t[6]+v)
2354     pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2355     pdf.refobj(on)
2356   end
2357 else
2358   pdfetcs.shadingpatterns = { }
2359   pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2360   function luamplib.dolatelua (on, kind, xobj)
2361     local h, v
2362     local t = pdfetcs.shadingpatterns[on] or { }
2363     local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2364       or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2365     if shift then
2366       h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2367     else
2368       h, v = pdf.getpos()
2369       h = format("%f", h/factor) :gsub(decimals,rmzeros)
2370       v = format("%f", v/factor) :gsub(decimals,rmzeros)
2371     end
2372     if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2373       warn"Rerun to get correct shading pattern"
2374     end
2375     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2376     local init = pdfetcs.shadingpatterninit_w
2377     if init then pdfetcs.shadingpatterninit_w = nil end
2378     local f = ioopen(name, init and "w" or "a")
2379     if f then
2380       f:write(("<<%s %s %s\n"):format(on, h, v))
2381       f:close()
2382     else
2383       err"cannot write a file. check the cache dir path"
2384     end
2385   end

```



```

2386 end
2387 local function do_preobj_shading (object, prescript)
2388   if not prescript or not prescript.sh_operand_type then return end
2389   local on = do_preobj_SH(object, prescript)
2390   local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2391   local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
2392   if matrix:find"%a" then
2393     local data = format("mplibtransformmatrix(%s);",matrix)
2394     process(data,"@mplibtransformmatrix")
2395     local t = luamplib.transformmatrix
2396     matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5], t[6]):gsub(decimals,rmzeros)
2397   end
2398   if prescript.sh_in_xobj == "yes" then
2399     on = update_pdfobjs(("<<%s/Matrix[%s]>>"):format(os, matrix))
2400     goto skip_latelua
2401   end
2402   on = update_pdfobjs()
2403   if pdfmode then
2404     put2output(tableconcat{"\\latelua{luamplib.dolatelua(",on,"",[",os,"],[",matrix,"]])}")
2405   else
2406     local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2407               or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2408     local init = pdfetcs.shadingpatterninit_r
2409     if init then
2410       pdfetcs.shadingpatterninit_r = nil
2411       local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2412       local f = ioopen(name)
2413       if f then
2414         for line in f:lines() do
2415           local t = line:explode()
2416           pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2417         end
2418         f:close()
2419       end
2420     end
2421   end

```

This seems to be needed for proper functioning:

```

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

2421   local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2422   local mt = matrix:explode()
2423   matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2424   texsprint{ "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2425             format(" <<%s/Matrix[%s]>>", os, matrix) }
2426   put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2427             xobj and ("%s',[[%s]]"):format(xobj[1], xobj[2]))
2428 end
2429 ::skip_latelua::

```

```

2430 local key, val = format("MPLibPt%s", on), format(pdfetcs.resfmt, on)
2431 add_pattern_resources(key, val)
2432 pdf_literalcode("/Pattern cs/%s scn", key)

```

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

```

2433 prescript.sh_type = nil
2434 end
2435

```

Tiling Patterns

```

2436 pdfetcs.patterns = { }
2437 local function gather_resources (optres, ispattern)
2438   local t = { }
2439   if pdfmanagement then
2440     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2441       local mytoks
2442       run_tex_code ( {
2443         "\\mplibtmptoks\\expanded{ {",
2444         "\\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/", v, "}",
2445         "{\\pdfdict_use:n{g__pdf_Core/Page/Resources/", v, "}}", " }",
2446       }, ccexplat )
2447       mytoks = texgettoks"mplibtmptoks"
2448       if not pdfmode then
2449         mytoks = mytoks:gsub("\\str_convert_pdfname:n{s*{(.-)})", "%1") -- why not expanded?
2450       end
2451       mytoks = mytoks and mytoks:gsub("^s*(.)s*$", "%1")
2452       if mytoks and mytoks ~= "" then
2453         t[#t+1] = ("%s<<s>>"):format(v, mytoks)
2454       end
2455     end
2456   elseif is_defined(pdfetcs.pgftxtgs) then
2457     run_tex_code "\\relax" -- flush tex.sprint queue
2458     if pdfmode then
2459       for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2460                         ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2461                         Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2462         local res = (get_macro(v) or ""):gsub("^s*(.)s*$", "%1")
2463         if res ~= "" then
2464           t[#t+1] = ("%s<<s>>"):format(k, res )
2465         end
2466       end
2467     else
2468       local abc = get_macro"pgfutil@abc" or ""
2469       for k,v in pairs { ExtGState = "@pgfextgs",
2470                         ColorSpace = "@pgfcolorspaces",
2471                         Pattern = "@pgfpatterns", } do
2472         local tt = { }
2473         for vv in abc:gmatch( v .. "s*(%b<>)" ) do
2474           tt[#tt+1] = vv:match("^<<s*(.)s*>>$")
2475         end

```

```

2476     if #tt > 0 then
2477         t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2478     end
2479 end
2480 end

```

We still have to deal with Shading resources.

```

2481 if luatexbase.callbacktypes.finish_pdffile then
2482     if pdfetcs.Shading_res then
2483         t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2484     end
2485 else
2486     local res = pdfetcs.getpageres()
2487     res = res and res:match"/Shading%s*%b<>"
2488     if res then
2489         t[#t+1] = res
2490     end
2491 end
2492 else
2493     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

```

2494     warn"transparent package is not fully functional without pdfmanagement code."
2495 end
2496 if luatexbase.callbacktypes.finish_pdffile then
2497     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2498         local tt = pdfetcs[v.."_res"]
2499         if tt then
2500             t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2501         end
2502     end
2503 else
2504     local res = pdfetcs.getpageres()
2505     if res then
2506         t[#t+1] = res
2507     end
2508 end
2509 end
2510 local result = tableconcat(t)
2511 if optres ~= "" then
2512     for _,v in ipairs { "ExtGState", "ColorSpace", "Pattern", "Shading" } do
2513         local res = optres:match("/"..v.."%"s*%b<>")
2514         if res then
2515             if result:find("/"..v) then
2516                 res = res:match("<<(.+)>>$")
2517                 result = result:gsub("/"..v.."%"s*<< ", "%1"..res, 1)
2518             else
2519                 result = result .. res
2520             end

```

```

2521     end
2522     end
2523 end
2524 return result
2525 end
2526 function luamplib.registerpattern ( boxid, name, opts )
2527   local box = texgetbox(boxid)
2528   local wd = format("%.3f",box.width/factor)
2529   local hd = format("%.3f", (box.height+box.depth)/factor)
2530   info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2531   if opts.xstep == 0 then opts.xstep = nil end
2532   if opts.ystep == 0 then opts.ystep = nil end
2533   if opts.colored == nil then
2534     opts.colored = opts.coloured
2535     if opts.colored == nil then
2536       opts.colored = true
2537     end
2538   end
2539   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2540   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2541   if opts.matrix and opts.matrix:find"%a" then
2542     local data = format("mplibtransformmatrix(%s);",opts.matrix)
2543     process(data,"@mplibtransformmatrix")
2544     local t = luamplib.transformmatrix
2545     opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2546     opts.xshift = opts.xshift or format("%f",t[5])
2547     opts.yshift = opts.yshift or format("%f",t[6])
2548   end
2549   local attr = {
2550     "/Type/Pattern",
2551     "/PatternType 1",
2552     format("/PaintType %i", opts.colored and 1 or 2),
2553     "/TilingType 2",
2554     format("/XStep %s", opts.xstep or wd),
2555     format("/YStep %s", opts.ystep or hd),
2556     format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2557   }
2558   local optres = opts.resources or ""
2559   optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2560   local patterns = pdfetcs.patterns
2561   if pdfmode then
2562     if opts.bbox then
2563       attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2564     end
2565     attr = tableconcat(attr) :gsub(decimals,rmzeros)
2566     local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2567     patterns[name] = { id = index, colored = opts.colored }
2568   else
2569     local cnt = #patterns + 1

```

```

2570 local objname = "@mplibpattern" .. cnt
2571 local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2572 texsprint {
2573   "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2574   "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2575   "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2576   "\\special{pdf:bcontent}",
2577   "\\special{pdf:bxobj ", objname, " ", metric, "}",
2578   "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2579   "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2580   "\\special{pdf:put @resources <<", optres, ">>}",
2581   "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2582   "\\special{pdf:econtent}}",
2583 }
2584 patterns[cnt] = objname
2585 patterns[name] = { id = cnt, colored = opts.colored }
2586 patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
2587 end
2588 end
2589
2590 local do_preobj_PAT
2591 do
2592   local function pattern_colorspace (cs)
2593     local on, new = update_pdfobjs(format("[Pattern %s]", cs))
2594     if new then
2595       local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
2596       if pdfmanagement then
2597         texsprint {
2598           "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
2599         }
2600       else
2601         local res = format("/%s %s", key, val)
2602         if is_defined(pdfetcs.pgfcolorspace) then
2603           texsprint { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{", res, "}" }
2604         else
2605           pdfetcs.fallback_update_resources("ColorSpace",res,"@MPlibCS")
2606         end
2607       end
2608     end
2609     return on
2610   end
2611   function do_preobj_PAT(object, prescript)
2612     local name = prescript and prescript.mplibpattern
2613     if not name then return end
2614     local patterns = pdfetcs.patterns
2615     local patt = patterns[name]
2616     local index = patt and patt.id or err("cannot get pattern object '%s'", name)
2617     local key = format("MPlibPt%s",index)
2618     if patt.colored then

```

```

2619     pdf_literalcode("/Pattern cs /%s scn", key)
2620 else
2621     local color = prescript.mpliboverridecolor
2622     if not color then
2623         local t = object.color
2624         color = t and #t>0 and luamplib.colorconverter(t)
2625     end
2626     if not color then return end
2627     local cs
2628     if color:find" cs " or color:find"@pdf.obj" then
2629         local t = color:explode()
2630         if pdfmode then
2631             cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
2632             color = t[3]
2633         else
2634             cs = t[2]
2635             color = t[3]:match"%[(.+)%"
2636         end
2637     else
2638         local t = colorsplit(color)
2639         cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
2640         color = tableconcat(t, " ")
2641     end
2642     pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
2643 end
2644 if not patt.done then
2645     local val = pdfmode and format("%s 0 R",index) or patterns[index]
2646     add_pattern_resources(key,val)
2647 end
2648 patt.done = true
2649 end
2650 end
2651

```

Fading

```

2652 pdfetcs.fading = { }
2653 local function do_preobj_FADE (object, prescript)
2654     local fd_type = prescript and prescript.mplibfadetype
2655     local fd_stop = prescript and prescript.mplibfadestate
2656     if not fd_type then
2657         return fd_stop -- returns "stop" (if picture) or nil
2658     end
2659     local on, os, new
2660     if fd_type == "masking" then
2661         local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
2662         on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
2663         local bc = prescript.mplibmaskingbgcolor
2664         bc = bc and bc:gsub(":", " ")
2665         bc = bc and ("BC[%s]"):format(bc):gsub(decimals,rmzeros) or ""

```

```

2666     os = format("<</SMask<</S/Luminosity/G %s%>>>>",
2667                 pdfmode and format(pdfetcs.resfmt, on) or on, bc)
2668 else
2669     local bbox = prescript.mplibfadebbox:explode":""
2670     local dx, dy = -bbox[1], -bbox[2]
2671     local vec = prescript.mplibfadevector; vec = vec and vec:explode":""
2672     if not vec then
2673         if fd_type == "linear" then
2674             vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
2675         else
2676             local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
2677             vec = {centerx, centery, centerx, centery} -- center for both circles
2678         end
2679     end
2680     local coords = { vec[1], vec[2], vec[3], vec[4] }
2681     if fd_type == "linear" then
2682         coords = format("%f %f %f %f", tableunpack(coords))
2683     elseif fd_type == "circular" then
2684         local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
2685         local radius = (prescript.mplibfaderadius or "0":..math.sqrt(width^2+height^2)/2):explode":""
2686         tableinsert(coords, 3, radius[1])
2687         tableinsert(coords, radius[2])
2688         coords = format("%f %f %f %f %f %f", tableunpack(coords))
2689     else
2690         err("unknown fading method '%s'", fd_type)
2691     end
2692     fd_type = fd_type == "linear" and 2 or 3
2693     local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
2694     local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":"" }
2695     local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":"" }
2696     if steps > 1 then
2697         fractions = { prescript.sh_fraction_1 or 0 }
2698         for i=2,steps do
2699             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2700             ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":""
2701             cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":""
2702         end
2703     end
2704     local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
2705     if matrix:find"%a" then
2706         local data = format("mplibtransformmatrix(%s);",matrix)
2707         process(data,"@mplibtransformmatrix")
2708         matrix = luamplib.transformmatrix
2709     else
2710         matrix = matrix:explode()
2711     end
2712     matrix[5] = matrix[5] + dx
2713     matrix[6] = matrix[6] + dy
2714     matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)

```

```

2715 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
2716 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
2717 on = update_pdfobjs(os)
2718 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
2719 local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
2720 :gsub(decimals,rmzeros)
2721 os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
2722 on = update_pdfobjs(os)
2723 local resources = format(pdfetcs.resfmt, on)
2724 on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
2725 local attr = tableconcat{
2726     "/Subtype/Form",
2727     "/BBox[" .. bbox .. "]",
2728     "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
2729     "/Resources " .. resources,
2730     "/Group " .. format(pdfetcs.resfmt, on),
2731 } :gsub(decimals,rmzeros)
2732 on = update_pdfobjs(attr, streamtext)
2733 os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
2734 end
2735 on, new = update_pdfobjs(os)
2736 local key = add_extgs_resources(on,new)
2737 start_pdf_code()
2738 pdf_literalcode("/%s gs", key)
2739 if fd_stop then return "standalone" end
2740 return "start"
2741 end
2742

```

Transparency Group

```

2743 pdfetcs.tr_group = { shifts = { } }
2744 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
2745 local function do_preobj_GRP (object, prescript)
2746     local grstate = prescript and prescript.gr_state
2747     if not grstate then return end
2748     local trgroup = pdfetcs.tr_group
2749     if grstate == "start" then
2750         trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
2751         trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
2752         for _,v in ipairs(prescript.gr_type:explode",+") do
2753             trgroup[v] = true
2754         end
2755         trgroup.bbox = prescript.mplibgroupbbox:explode": "
2756         put2output[["\beginpgroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
2757     elseif grstate == "stop" then
2758         local llx,lly,urx,ury = tableunpack(trgroup.bbox)
2759         put2output(tableconcat{
2760             "\\egroup",
2761             format("\\wd\mplibscratchbox %fbp", urx-llx),

```



```

2762     format("\ht\mplibscratchbox %fbp", ury-lly),
2763     "\dp\mplibscratchbox 0pt",
2764   })
2765   local grattr
2766   if trgroup.off then
2767     grattr = ""
2768   else
2769     local on = update_pdfobjs(format("</S/Transparency/I %s/K %s>>",
2770                                     trgroup.isolated, trgroup.knockout))
2771     grattr = format("/Group %s", pdfetcs.resfmt:format(on))
2772   end
2773   local res = gather_resources("")
2774   local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
2775   if pdfmode then
2776     put2output(tableconcat{
2777       "\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
2778       "/BBox[" .. bbox .. "], grattr, "} resources{" .. res .. "}" .. "\mplibscratchbox",
2779       "\luamplibtagasgroupput{" .. trgroup.name .. "}" ..
2780       "[[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]]",
2781       "[[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]]",
2782       "[[\box\mplibscratchbox]]",
2783       "}" .. "\endgroup",
2784       "\expandafter\edef\csname luamplib.group.", trgroup.name, "\endcsname{" ..
2785       "\setbox\mplibscratchbox\hbox{\hskip", -llx, "bp\raise", -lly, "bp\hbox{" ..
2786       "\useboxresource \the\lastsavedboxresourceindex",
2787       "}}\wd\mplibscratchbox", urx-llx, "bp\ht\mplibscratchbox", ury-lly, "bp",
2788       "\box\mplibscratchbox}",
2789     })
2790   else
2791     trgroup.cnt = (trgroup.cnt or 0) + 1
2792     local objname = format("@mplibtrgr%s", trgroup.cnt)
2793     put2output(tableconcat{
2794       "\special{pdf:bxobj " .. objname .. " bbox " .. bbox .. "}",
2795       "\unhbox\mplibscratchbox",
2796       "\special{pdf:put @resources <<", res, ">>}",
2797       "\special{pdf:exobj <<", grattr, ">>}",
2798       "\luamplibtagasgroupput{" .. trgroup.name .. "}" ..
2799       "\special{pdf:uxobj " .. objname .. "}",
2800       "}" .. "\endgroup",
2801     })
2802     token.set_macro("luamplib.group." .. trgroup.name, tableconcat{
2803       "\setbox\mplibscratchbox\hbox{\hskip", -llx, "bp\raise", -lly, "bp\hbox{" ..
2804       "\special{pdf:uxobj " .. objname .. "}",
2805       "}}\wd\mplibscratchbox", urx-llx, "bp\ht\mplibscratchbox", ury-lly, "bp",
2806       "\box\mplibscratchbox",
2807     }, "global")
2808   end
2809   trgroup.shifts[trgroup.name] = { llx, lly }
2810 end

```

```

2811 return grstate
2812 end
2813 function luamplib.registergroup (boxid, name, opts)
2814   if opts.asgroup and opts.asgroup:find"wrapped" then
2815     luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
2816     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}" }
2817     opts.asgroup = opts.asgroup:gsub("wrapped", "")
2818   end
2819   local box = texgetbox(boxid)
2820   local wd, ht, dp = node.getwhd(box)
2821   local is_mask = opts.asgroup and opts.asgroup:find"masking"
2822   local res = opts.resources or ""
2823   res = gather_resources(res)
2824   local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
2825   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2826   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2827   if opts.matrix and opts.matrix:find"%a" then
2828     local data = format("mplibtransformmatrix(%s);", opts.matrix)
2829     process(data, "@mplibtransformmatrix")
2830     opts.matrix = format("%f %f %f %f %f %f", tableunpack(luamplib.transformmatrix))
2831   end
2832   local grtype = 3
2833   if opts.bbox then
2834     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2835     grtype = 2
2836   end
2837   local mpllx, mpily = get_macro'MPlly', get_macro'MPlly'
2838   if is_mask then
2839     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
2840     t[5], t[6] = t[5]+mpllx, t[6]+mpily
2841     opts.matrix = format("%f %f %f %f %f %f", tableunpack(t))
2842     mpllx, mpily = 0, 0
2843   end
2844   if opts.matrix then
2845     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
2846     grtype = opts.bbox and 4 or 1
2847   end
2848   if opts.asgroup and not opts.asgroup:find"off" then
2849     local t = { isolated = false, knockout = false, masking = false }
2850     for _, v in ipairs(opts.asgroup:explode",") do t[v] = true end
2851     local on
2852     if t.masking then
2853       on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorsname or "/DeviceGray"))
2854     else
2855       local cs = opts.colorsname and ("/CS%s"):format(opts.colorsname) or ""
2856       on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
2857     end
2858     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
2859   end

```

```

2860 local trgroup = pdfetcs.tr_group
2861 trgroup.shifts[name] = { mpllx, mplly }
2862 local whd
2863 if pdfmode then
2864   attr = tableconcat(attr) :gsub(decimals,rmzeros)
2865   local index = tex.saveboxresource(boxid, attr, res, true, grtype)
2866   token.set_macro("luamplib.group"..name, tableconcat{
2867     "\\useboxresource ", index,
2868   }, "global")
2869   whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
2870 else
2871   trgroup.cnt = (trgroup.cnt or 0) + 1
2872   local objname = format("@mplibtrgr%s", trgroup.cnt)
2873   textsprint {
2874     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2875     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2876     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2877     "\\special{pdf:bcontent}",
2878     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
2879     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2880     "\\special{pdf:put @resources <<", res, ">>}",
2881     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2882     "\\special{pdf:econtent}}",
2883   }
2884   token.set_macro("luamplib.group"..name, tableconcat{
2885     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
2886     "\\wd\\mplibscratchbox ", wd, "sp",
2887     "\\ht\\mplibscratchbox ", ht, "sp",
2888     "\\dp\\mplibscratchbox ", dp, "sp",
2889     "\\box\\mplibscratchbox",
2890   }, "global")
2891   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
2892 end
2893 info("w/h/d of group '%s': %s", name, whd)
2894 end
2895

```

luamplib.convert: flushing figures

```

2896 do
2897   local function stop_special_effects(fade,opaq,over)
2898     if fade then -- fading
2899       stop_pdf_code()
2900     end
2901     if opaq then -- opacity
2902       pdf_literalcode(opaq)
2903     end
2904     if over then -- color
2905       if over:find"pdf:bc" then
2906         put2output"\\special{pdf:ec}"

```

```

2907     else
2908         put2output"\special{color pop}"
2909     end
2910 end
2911 end
2912

```

For parsing prescript materials.

```

2913 local function script2table(s)
2914     local t = {}
2915     for _,i in ipairs(s:explode("\13+")) do
2916         local k,v = i:match("(.)=(.*)") -- v may contain = or empty.
2917         if k and v and k ~= "" and not t[k] then
2918             t[k] = v
2919         end
2920     end
2921     return t
2922 end
2923

```

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

```

2924 local function pdf_textfigure(font,size,text,width,height,depth)
2925     text = text:gsub(".",function(c)
2926         return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
2927     end)
2928     put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
2929 end
2930
2931 local bend_tolerance = 131/65536
2932
2933 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
2934
2935 local function pen_characteristics(object)
2936     local t = mplib.pen_info(object)
2937     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
2938     divider = sx*sy - rx*ry
2939     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
2940 end
2941
2942 local function concat(px, py) -- no tx, ty here
2943     return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
2944 end
2945
2946 local function curved(ith,pth)
2947     local d = pth.left_x - ith.right_x
2948     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
2949         abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
2950         d = pth.left_y - ith.right_y
2951         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and

```

```

2952         abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
2953         return false
2954     end
2955 end
2956 return true
2957 end
2958
2959 local function flushnormalpath(path,open)
2960     local pth, ith
2961     for i=1,#path do
2962         pth = path[i]
2963         if not ith then
2964             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
2965         elseif curved(ith,pth) then
2966             pdf_literalcode("%f %f %f %f %f %f c",
2967                 ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
2968         else
2969             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
2970         end
2971         ith = pth
2972     end
2973     if not open then
2974         local one = path[1]
2975         if curved(pth,one) then
2976             pdf_literalcode("%f %f %f %f %f %f c",
2977                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
2978         else
2979             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2980         end
2981     elseif #path == 1 then -- special case .. draw point
2982         local one = path[1]
2983         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2984     end
2985 end
2986
2987 local function flushconcatpath(path,open)
2988     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
2989     local pth, ith
2990     for i=1,#path do
2991         pth = path[i]
2992         if not ith then
2993             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
2994         elseif curved(ith,pth) then
2995             local a, b = concat(ith.right_x,ith.right_y)
2996             local c, d = concat(pth.left_x,pth.left_y)
2997             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
2998         else
2999             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3000     end

```

```

3001     ith = pth
3002 end
3003 if not open then
3004     local one = path[1]
3005     if curved(pth,one) then
3006         local a, b = concat(pth.right_x,pth.right_y)
3007         local c, d = concat(one.left_x,one.left_y)
3008         pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3009     else
3010         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3011     end
3012 elseif #path == 1 then -- special case .. draw point
3013     local one = path[1]
3014     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3015 end
3016 end
3017

```

Finally, flush figures by inserting PDF literals.

```

3018 local function flush (result,flusher)
3019     if result then
3020         local figures = result.fig
3021         if figures then
3022             for f=1, #figures do
3023                 info("flushing figure %s",f)
3024                 local figure = figures[f]
3025                 local objects = figure:objects()
3026                 local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
3027                 local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3028                 local bbox = figure:boundingbox()
3029                 local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3030                 if urx < llx then

```

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

```

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

```

```

3031     else

```

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

```

3032         if tex_code_pre_mplib[f] then
3033             put2output(tex_code_pre_mplib[f])
3034         end
3035         pdf_startfigure(fignum,llx,lly,urx,ury)
3036         start_pdf_code()
3037         if objects then
3038             local savedpath = nil
3039             local savedhtap = nil

```

```

3040         for o=1,#objects do
3041             local object      = objects[o]
3042             local objecttype  = object.type

```

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

```

3043         local prescript      = object.prescript
3044         prescript = prescript and script2table(prescript) -- prescript is now a table
3045         local cr_over = do_preobj_CR(object,prescript) -- color
3046         local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3047         local fading_ = do_preobj_FADE(object,prescript) -- fading
3048         local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
3049         local shading_ = do_preobj_shading(object,prescript) -- shading pattern
3050         local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3051         if prescript and prescript.mplibtexboxid then
3052             put_tex_boxes(object,prescript)
3053         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3054             elseif objecttype == "start_clip" then
3055                 local evenodd = not object.istext and object.postscript == "evenodd"
3056                 start_pdf_code()
3057                 flushnormalpath(object.path,false)
3058                 pdf_literalcode(evenodd and "W* n" or "W n")
3059             elseif objecttype == "stop_clip" then
3060                 stop_pdf_code()
3061                 miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3062             elseif objecttype == "special" then

```

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

```

3063         if prescript and prescript.postmplibverbtx then
3064             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
3065         end
3066         elseif objecttype == "text" then
3067             local ot = object.transform -- 3,4,5,6,1,2
3068             start_pdf_code()
3069             pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3070             pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3071             stop_pdf_code()
3072         elseif not trgroup and fading_ ~= "stop" then
3073             local evenodd, collect, both = false, false, false
3074             local postscript = object.postscript
3075             if not object.istext then
3076                 if postscript == "evenodd" then
3077                     evenodd = true
3078                 elseif postscript == "collect" then
3079                     collect = true
3080                 elseif postscript == "both" then
3081                     both = true
3082                 elseif postscript == "eoboth" then
3083                     evenodd = true
3084                     both = true
3085             end

```

```

3086         end
3087     if collect then
3088         if not savedpath then
3089             savedpath = { object.path or false }
3090             savedhtap = { object.htap or false }
3091         else
3092             savedpath[#savedpath+1] = object.path or false
3093             savedhtap[#savedhtap+1] = object.htap or false
3094         end
3095     else

```

Removed from ConTeXt general: color stuff.

```

3096         local ml = object.miterlimit
3097         if ml and ml ~= miterlimit then
3098             miterlimit = ml
3099             pdf_literalcode("%f M",ml)
3100         end
3101         local lj = object.linejoin
3102         if lj and lj ~= linejoin then
3103             linejoin = lj
3104             pdf_literalcode("%i j",lj)
3105         end
3106         local lc = object.linecap
3107         if lc and lc ~= linecap then
3108             linecap = lc
3109             pdf_literalcode("%i J",lc)
3110         end
3111         local dl = object.dash
3112         if dl then
3113             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3114             if d ~= dashed then
3115                 dashed = d
3116                 pdf_literalcode(dashed)
3117             end
3118         elseif dashed then
3119             pdf_literalcode("[ ] 0 d")
3120             dashed = false
3121         end
3122         local path = object.path
3123         local transformed, penwidth = false, 1
3124         local open = path and path[1].left_type and path[#path].right_type
3125         local pen = object.pen
3126         if pen then
3127             if pen.type == 'elliptical' then
3128                 transformed, penwidth = pen_characteristics(object) -- boolean, value
3129                 pdf_literalcode("%f w",penwidth)
3130                 if objecttype == 'fill' then
3131                     objecttype = 'both'
3132                 end

```



```

3133         else -- calculated by mplib itself
3134             objecttype = 'fill'
3135         end
3136     end

```

Added : shading

```

3137     local shade_no, shade_stroking = do_preobj_SH(object,prescript) -- shading
3138     if shade_no then
3139         pdf_literalcode"q /Pattern cs"
3140         objecttype = false
3141     end
3142     if transformed then
3143         start_pdf_code()
3144     end
3145     if path then
3146         if savedpath then
3147             for i=1,#savedpath do
3148                 local path = savedpath[i]
3149                 if transformed then
3150                     flushconcatpath(path,open)
3151                 else
3152                     flushnormalpath(path,open)
3153                 end
3154             end
3155             savedpath = nil
3156         end
3157         if transformed then
3158             flushconcatpath(path,open)
3159         else
3160             flushnormalpath(path,open)
3161         end
3162         if objecttype == "fill" then
3163             pdf_literalcode(evenodd and "h f*" or "h f")
3164         elseif objecttype == "outline" then
3165             if both then
3166                 pdf_literalcode(evenodd and "h B*" or "h B")
3167             else
3168                 pdf_literalcode(open and "S" or "h S")
3169             end
3170         elseif objecttype == "both" then
3171             pdf_literalcode(evenodd and "h B*" or "h B")
3172         end
3173     end
3174     if transformed then
3175         stop_pdf_code()
3176     end
3177     local path = object.htap

```

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

```

3178     if path then

```

```

3179         if transformed then
3180             start_pdf_code()
3181         end
3182         if savedhtap then
3183             for i=1,#savedhtap do
3184                 local path = savedhtap[i]
3185                 if transformed then
3186                     flushconcatpath(path,open)
3187                 else
3188                     flushnormalpath(path,open)
3189                 end
3190             end
3191             savedhtap = nil
3192             evenodd = true
3193         end
3194         if transformed then
3195             flushconcatpath(path,open)
3196         else
3197             flushnormalpath(path,open)
3198         end
3199         if objecttype == "fill" then
3200             pdf_literalcode(evenodd and "h f*" or "h f")
3201         elseif objecttype == "outline" then
3202             pdf_literalcode(open and "S" or "h S")
3203         elseif objecttype == "both" then
3204             pdf_literalcode(evenodd and "h B*" or "h B")
3205         end
3206         if transformed then
3207             stop_pdf_code()
3208         end
3209     end

```

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

```

3210         if shade_no then -- shading
3211             pdf_literalcode("W%s %s /MPLibSh%s sh Q",
3212                 evenodd and "*" or "", shade_stroking and "s" or "n", shade_no)
3213         end
3214     end
3215 end
3216 if fading_ == "start" then
3217     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}
3218 elseif trgroup == "start" then
3219     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
3220 elseif fading_ == "stop" then
3221     local se = pdfetcs.fading.specialeffects
3222     if se then stop_special_effects(se[1], se[2], se[3]) end
3223 elseif trgroup == "stop" then
3224     local se = pdfetcs.tr_group.specialeffects
3225     if se then stop_special_effects(se[1], se[2], se[3]) end

```

```

3226         else
3227             stop_special_effects(fading_, tr_opaq, cr_over)
3228         end
3229         if fading_ or trgroup then -- extgs resetted
3230             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3231         end
3232     end
3233 end
3234 stop_pdf_code()
3235 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimtexp code.

```

3236         for _,v in ipairs(figcontents) do
3237             if type(v) == "table" then
3238                 texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3239             else
3240                 texsprint(v)
3241             end
3242         end
3243         if #figcontents.post > 0 then texsprint(figcontents.post) end
3244         figcontents = { post = { } }
3245     end
3246 end
3247 end
3248 end
3249 end
3250
3251 function luamplib.convert (result, flusher)
3252     flush(result, flusher)
3253     return true -- done
3254 end
3255 end
3256
3257 function luamplib.colorconverter (cr)
3258     local n = #cr
3259     if n == 4 then
3260         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3261         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3262     elseif n == 3 then
3263         local r, g, b = cr[1], cr[2], cr[3]
3264         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3265     else
3266         local s = cr[1]
3267         return format("%.3f g %.3f G",s,s), "0 g 0 G"
3268     end
3269 end

```

2.2 T_EX package

First we need to load some packages.

```
3270 \ifcsname ProvidesPackage\endcsname
```

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

```
3271 \NeedsTeXFormat{LaTeX2e}
3272 \ProvidesPackage{luamplib}
3273 [2026/05/12 v2.41.2 mplib package for LuaTeX]
3274 \fi
3275 \ifdefined\newluafunction\else
3276 \input ltluatex
3277 \fi
```

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

```
3278 \ifnum\outputmode=0
3279 \ifdefined\AddToHookNext
3280 \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3281 \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3282 \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3283 \else
3284 \input atbegshi.sty
3285 \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3286 \let\luamplibatfirstshipout\AtBeginShipoutFirst
3287 \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
3288 \fi
3289 \fi
```

Loading of lua code.

```
3290 \directlua{require("luamplib")}
```

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

```
3291 \ifx\pdfoutput\undefined
3292 \let\pdfoutput\outputmode
3293 \fi
3294 \ifx\pdfliteral\undefined
3295 \protected\def\pdfliteral{\pdfextension literal}
3296 \fi
```

Set the format for METAPOST.

```
3297 \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.

```
3298 \ifnum\pdfoutput>0
3299 \let\mplibtoPDF\pdfliteral
3300 \else
```

```

3301 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
3302 \ifcsname PackageInfo\endcsname
3303   \PackageInfo{luamplib}{only dvipdfmx is supported currently}
3304 \else
3305   \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
3306 \fi
3307 \fi

```

To make mplibcode typeset always in horizontal mode.

```

3308 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
3309 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
3310 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

3311 \def\mplibsetupcatcodes{%
3312   %catcode`\{=12 %catcode`\}=12
3313   \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
3314   \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^^M=12
3315 }

```

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

```

3316 \ifnum\outputmode>0 %
3317   \def\mplibputtextbox#1#2#3#4{%
3318     \pdfextension save\relax
3319     \vbox to 0pt{\vss
3320       \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
3321       \kern#3bp}%
3322     \pdfextension restore\relax}
3323 \else
3324   \def\mplibputtextbox#1#2#3#4{%
3325     \special{pdf:btrans matrix #4 #2 #3}%
3326     \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
3327     \special{pdf:etrans}}
3328 \fi

```

use Transparency Group

```

3329 \protected\def\usemplibgroup#1#2{\usemplibgroupmain}
3330 \def\usemplibgroupmain#1{%
3331   \prependtomplibbox\hbox dir TLT\bgroup
3332   \csname luamplib.group.#1\endcsname
3333   \egroup
3334 }
3335 \protected\def\mplibgroup#1{%
3336   \begingroup
3337   \def\MPllx{0}\def\MPlly{0}%
3338   \def\mplibgroupname{#1}%
3339   \mplibgroupgetnexttok
3340 }
3341 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3342 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok= }

```

```

3343 \def\mplibgroupbranch{%
3344   \ifx [\nexttok
3345     \expandafter\mplibgroupopts
3346   \else
3347     \ifx\mplibsptoken\nexttok
3348       \expandafter\expandafter\expandafter\mplibgroupskipspace
3349     \else
3350       \let\mplibgroupoptions\empty
3351     \expandafter\expandafter\expandafter\mplibgroupmain
3352   \fi
3353 \fi
3354 }
3355 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3356 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3357 \protected\def\endmplibgroup{\egroup
3358   \directlua{ luamplib.registergroup(
3359     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3360   )}%
3361 \endgroup
3362 }

```

Patterns

```

3363 {\def\:{\global\let\mplibsptoken= } \: }
3364 \protected\def\mppattern#1{%
3365   \begingroup
3366   \def\MP11x{0}\def\MP11y{0}%
3367   \def\mplibpatternname{#1}%
3368   \mplibpatterngetnexttok
3369 }
3370 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3371 \def\mplibpatternskipspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
3372 \def\mplibpatternbranch{%
3373   \ifx [\nexttok
3374     \expandafter\mplibpatternopts
3375   \else
3376     \ifx\mplibsptoken\nexttok
3377       \expandafter\expandafter\expandafter\mplibpatternskipspace
3378     \else
3379       \let\mplibpatternoptions\empty
3380     \expandafter\expandafter\expandafter\mplibpatternmain
3381   \fi
3382 \fi
3383 }
3384 \def\mplibpatternopts[#1]{%
3385   \def\mplibpatternoptions{#1}%
3386   \mplibpatternmain
3387 }
3388 \def\mplibpatternmain{%
3389   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces

```

```

3390 }
3391 \protected\def\endmppattern{%
3392   \egroup
3393   \directlua{ luamplib.registerpattern(
3394     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3395   )}%
3396   \endgroup
3397 }

```

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

```

3398 \def\mpfiginstancename{@mpfig}
3399 \protected\def\mpfig{%
3400   \begingroup
3401   \futurelet\nexttok\mplibmpfigbranch
3402 }
3403 \def\mplibmpfigbranch{%
3404   \ifx *\nexttok
3405     \expandafter\mplibprempfig
3406   \else
3407     \ifx [\nexttok
3408       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3409     \else
3410       \expandafter\expandafter\expandafter\mplibmainmpfig
3411     \fi
3412   \fi
3413 }
3414 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
3415 \def\mplibmainmpfig{%
3416   \begingroup
3417   \mplibsetupcatcodes
3418   \mplibdomainmpfig
3419 }
3420 \long\def\mplibdomainmpfig#1\endmpfig{%
3421   \endgroup
3422   \directlua{
3423     local legacy = luamplib.legacyverbatimimtx
3424     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3425     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3426     luamplib.legacyverbatimimtx = false
3427     luamplib.everymplib["\mpfiginstancename"] = ""
3428     luamplib.everyendmplib["\mpfiginstancename"] = ""
3429     luamplib.process_mplibcode(
3430       "beginfig(0) "..everympfig.." "..[===[\unexpanded{#1}]===".." "..everyendmpfig.." endfig;",
3431       "\mpfiginstancename")
3432     luamplib.legacyverbatimimtx = legacy
3433     luamplib.everymplib["\mpfiginstancename"] = everympfig
3434     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3435   }%
3436   \endgroup

```

```

3437 }
3438 \def\mplibprempfig#1{%
3439   \begingroup
3440   \mplibsetupcatcodes
3441   \mplibdoprempfig
3442 }
3443 \long\def\mplibdoprempfig#1\endmpfig{%
3444   \endgroup
3445   \directlua{
3446     local legacy = luamplib.legacyverbatim
3447     local everympfig = luamplib.everymplib["\mpfiginstancename"]
3448     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3449     luamplib.legacyverbatim = false
3450     luamplib.everymplib["\mpfiginstancename"] = ""
3451     luamplib.everyendmplib["\mpfiginstancename"] = ""
3452     luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
3453     luamplib.legacyverbatim = legacy
3454     luamplib.everymplib["\mpfiginstancename"] = everympfig
3455     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3456   }%
3457   \endgroup
3458 }
3459 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3460 \unless\ifcsname ver@luamplib.sty\endcsname
3461   \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3462   \protected\def\mplibcode{%
3463     \begingroup
3464     \futurelet\nexttok\mplibcodebranch
3465   }
3466   \def\mplibcodebranch{%
3467     \ifx [\nexttok
3468       \expandafter\mplibcodegetinstancename
3469     \else
3470       \global\let\currentmpinstancename\empty
3471       \expandafter\mplibcodeindeed
3472     \fi
3473   }
3474   \def\mplibcodeindeed{%
3475     \begingroup
3476     \mplibsetupcatcodes
3477     \mplibdocode
3478   }
3479   \long\def\mplibdocode#1\endmplibcode{%
3480     \endgroup
3481     \directlua{luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\currentmpinstancename")}%
3482   \endgroup
3483 }

```



```

3484 \protected\def\endmplibcode{\endmplibcode}
3485 \else

```

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

```

3486 \newenvironment{mplibcode}[1][{}]{%
3487   \xdef\currentmpinstancename{#1}%
3488   \mplibtmptoks{}\ltxdomplibcode
3489 }{}
3490 \def\ltxdomplibcode{%
3491   \begingroup
3492   \mplibsetupcatcodes
3493   \ltxdomplibcodeindeed
3494 }
3495 \def\mplib@mplibcode{mplibcode}
3496 \long\def\ltxdomplibcodeindeed#1\end#2{%
3497   \endgroup
3498   \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3499   \def\mplibtemp@a{#2}%
3500   \ifx\mplib@mplibcode\mplibtemp@a
3501     \directlua{luamplib.process_mplibcode([===\the\mplibtmptoks===],"\currentmpinstancename")}%
3502     \end{mplibcode}%
3503   \else
3504     \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3505     \expandafter\ltxdomplibcode
3506   \fi
3507 }
3508 \fi

```

User settings.

```

3509 \def\mplibshowlog#1{\directlua{
3510   local s = string.lower("#1")
3511   if s == "enable" or s == "true" or s == "yes" then
3512     luamplib.showlog = true
3513   else
3514     luamplib.showlog = false
3515   end
3516 }}
3517 \def\mpliblegacybehavior#1{\directlua{
3518   local s = string.lower("#1")
3519   if s == "enable" or s == "true" or s == "yes" then
3520     luamplib.legacyverbatimtex = true
3521   else
3522     luamplib.legacyverbatimtex = false
3523   end
3524 }}
3525 \def\mplibverbatim#1{\directlua{
3526   local s = string.lower("#1")
3527   if s == "enable" or s == "true" or s == "yes" then
3528     luamplib.verbatiminput = true
3529   else

```

```

3530     luamplib.verbatiminput = false
3531 end
3532 }}
3533 \newtoks\mplibmptoks

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

3534 \ifcsname ver@luamplib.sty\endcsname
3535 \protected\def\everymplib{%
3536   \begingroup
3537   \mplibsetupcatcodes
3538   \mplibdoeverymplib
3539 }
3540 \protected\def\everyendmplib{%
3541   \begingroup
3542   \mplibsetupcatcodes
3543   \mplibdoeveryendmplib
3544 }
3545 \newcommand\mplibdoeverymplib[2][]{%
3546   \endgroup
3547   \directlua{
3548     luamplib.everymplib["#1"] = [===[\unexpanded{#2}]==]
3549   }%
3550 }
3551 \newcommand\mplibdoeveryendmplib[2][]{%
3552   \endgroup
3553   \directlua{
3554     luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]==]
3555   }%
3556 }
3557 \else
3558 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3559 \protected\def\everymplib#1{%
3560   \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3561   \begingroup
3562   \mplibsetupcatcodes
3563   \mplibdoeverymplib
3564 }
3565 \long\def\mplibdoeverymplib#1{%
3566   \endgroup
3567   \directlua{
3568     luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]==]
3569   }%
3570 }
3571 \protected\def\everyendmplib#1{%
3572   \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3573   \begingroup
3574   \mplibsetupcatcodes
3575   \mplibdoeveryendmplib
3576 }

```

```

3577 \long\def\mplibdoeveryendmplib#1{%
3578   \endgroup
3579   \directlua{
3580     luampLib.everyendmplib["\currentmpinstancename"] = [==[\unexpanded{#1}]===]
3581   }%
3582 }
3583 \fi

```

TeX macros for dimen/color

```

3584 \def\mpdim#1{ \runscript{"luampLibdimen{#1}"} }
3585 \def\mpcolor#1#{\domplibcolor{#1}}
3586 \def\domplibcolor#1#2{ \runscript{"luampLibcolor{#1{#2}"} } }

```

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

```

3587 \def\mplibnumbersystem#1{\directlua{
3588   local t = "#1"
3589   if t == "binary" then t = "decimal" end
3590   luampLib.numbersystem = t
3591 }}

```

Settings for .mp cache files.

```

3592 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
3593 \def\mplibdomakenocache#1,{%
3594   \ifx\empty#1\empty
3595     \expandafter\mplibdomakenocache
3596   \else
3597     \ifx\stop#1\else
3598       \directlua{luampLib.noneedtoreplace["#1.mp"]=true}%
3599       \expandafter\expandafter\expandafter\mplibdomakenocache
3600     \fi
3601   \fi
3602 }
3603 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
3604 \def\mplibdocancelnocache#1,{%
3605   \ifx\empty#1\empty
3606     \expandafter\mplibdocancelnocache
3607   \else
3608     \ifx\stop#1\else
3609       \directlua{luampLib.noneedtoreplace["#1.mp"]=false}%
3610       \expandafter\expandafter\expandafter\mplibdocancelnocache
3611     \fi
3612   \fi
3613 }
3614 \def\mplibcachedir#1{\directlua{luampLib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

3615 \def\mplibtexttextlabel#1{\directlua{
3616   local s = string.lower("#1")
3617   if s == "enable" or s == "true" or s == "yes" then
3618     luampLib.texttextlabel = true

```

```

3619     else
3620         luamplib.texttextlabel = false
3621     end
3622 }}
3623 \def\mplibcodeinherit#1{\directlua{
3624     local s = string.lower("#1")
3625     if s == "enable" or s == "true" or s == "yes" then
3626         luamplib.codeinherit = true
3627     else
3628         luamplib.codeinherit = false
3629     end
3630 }}
3631 \def\mplibglobaltexttext#1{\directlua{
3632     local s = string.lower("#1")
3633     if s == "enable" or s == "true" or s == "yes" then
3634         luamplib.globaltexttext = true
3635     else
3636         luamplib.globaltexttext = false
3637     end
3638 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

3639 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

3640 \def\mplibstarttoPDF#1#2#3#4{%
3641     \prependtomplibbox
3642     \hbox dir TLT\bgroup
3643     \xdef\MPllx{#1}\xdef\MPlly{#2}%
3644     \xdef\MPurx{#3}\xdef\MPury{#4}%
3645     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3646     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3647     \parskip0pt%
3648     \leftskip0pt%
3649     \parindent0pt%
3650     \everypar{}%
3651     \setbox\mplibscratchbox\vbox\bgroup
3652     \noindent
3653 }
3654 \def\mplibstoptoPDF{%
3655     \par
3656     \egroup %
3657     \setbox\mplibscratchbox\hbox %
3658         {\hskip-\MPllx bp%
3659          \raise-\MPlly bp%
3660          \box\mplibscratchbox}%
3661     \setbox\mplibscratchbox\vbox to \MPheight
3662         {\vfill
3663          \hsize\MPwidth

```

```

3664 \wd\mplibscratchbox0pt%
3665 \ht\mplibscratchbox0pt%
3666 \dp\mplibscratchbox0pt%
3667 \box\mplibscratchbox}%
3668 \wd\mplibscratchbox\MPwidth
3669 \ht\mplibscratchbox\MPheight
3670 \box\mplibscratchbox
3671 \egroup
3672 }

```

Text items have a special handler.

```

3673 \def\mplibtexttext#1#2#3#4#5{%
3674 \begingroup
3675 \setbox\mplibscratchbox\hbox
3676 {\font\temp=#1 at #2bp%
3677 \temp
3678 #3}%
3679 \setbox\mplibscratchbox\hbox
3680 {\hskip#4 bp%
3681 \raise#5 bp%
3682 \box\mplibscratchbox}%
3683 \wd\mplibscratchbox0pt%
3684 \ht\mplibscratchbox0pt%
3685 \dp\mplibscratchbox0pt%
3686 \box\mplibscratchbox
3687 \endgroup
3688 }

```

Input luamplib.cfg when it exists.

```

3689 \openin0=luamplib.cfg
3690 \ifeof0 \else
3691 \closein0
3692 \input luamplib.cfg
3693 \fi

```

Code for tagpdf

```

3694 \def\luamplibtagtextboxset#1#2{#2}
3695 \let\luamplibnotagtextboxset\luamplibtagtextboxset
3696 \let\luamplibtagasgroupset\relax
3697 \let\luamplibtagasgroupput\luamplibtagtextboxset
3698 \ifcsname SuspendTagging\endcsname\else\endinput\fi
3699 \ifcsname ver@tagpdf.sty\endcsname \else
3700 \ExplSyntaxOn
3701 \keys_define:nn{luamplib/tagging}
3702 {
3703 ,alt .code:n = { }
3704 ,actualtext .code:n = { }
3705 ,artifact .code:n = { }
3706 ,text .code:n = { }
3707 ,off .code:n = { }

```

```

3708 ,tag .code:n = { }
3709 ,adjust-BBox .code:n = { }
3710 ,tagging-setup .code:n = { }
3711 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3712 ,instancename .meta:n = { instance = {#1} }
3713 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
3714 }
3715 \RenewDocumentCommand\mplibcode{0{}}
3716 {
3717 \tl_gclear:N \currentmpinstancename
3718 \keys_set:ne{luamplib/tagging}{#1}
3719 \mplibtmptoks{}\ltxdomplibcode
3720 }
3721 \cs_set_eq:NN \mplibaltext \use_none:n
3722 \cs_set_eq:NN \mplibactualtext \use_none:n

```

2025/12/05: `\begin{center}\mpfig ... \endmpfig\end{center}` raises an Error! as we issue `\everypar{}` before flushing literals out. It is related to `\partokencontext=2` recently introduced by L^AT_EX. Why we used `vbox` initially? where `hbox` seems to be sufficient. Anyway, among various solutions including `\partokencontext\z@`, `\let\par\@@par`, and `\endgraf`, we here attempt to address the issue by adding the following line, which L^AT_EX's `\everypar` should have done.

```

3723 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
3724 \ExplSyntaxOff
3725 \endinput\fi
3726 \ExplSyntaxOn
3727 \tl_new:N \l__luamplib_tag_envname_tl
3728 \tl_new:N \l__luamplib_tag_alt_tl
3729 \tl_new:N \l__luamplib_tag_alt_dflt_tl
3730 \tl_new:N \l__luamplib_tag_actual_tl
3731 \tl_new:N \l__luamplib_tag_struct_tl
3732 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
3733 \bool_new:N \l__luamplib_tag_usetext_bool
3734 \bool_new:N \l__luamplib_tag_bboxcorr_bool
3735 \seq_new:N \l__luamplib_tag_bboxcorr_seq
3736 \tl_new:N \l__luamplib_tag_bboxdraw_tl
3737 \tl_new:N \l__luamplib_BBox_llx_tl
3738 \tl_new:N \l__luamplib_BBox_lly_tl
3739 \tl_new:N \l__luamplib_BBox_urx_tl
3740 \tl_new:N \l__luamplib_BBox_ury_tl
3741 \msg_new:nnn {luamplib}{figure-text-reuse}
3742 {
3743 tex-text~box~#1~probably~is~incorrectly~tagged.~
3744 Reusing~a~box~in~text~mode~is~strongly~discouraged.~
3745 Check~the~resulting~PDF.
3746 }
3747 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
3748 {
3749 mplibgroup~'#1'~probably~is~incorrectly~tagged.~
3750 Using~mplibgroup~with~text~mode~is~not~recommended.~

```

```

3751 Check~the~resulting~PDF.
3752 }
3753 \msg_new:nnn{luamplib}{alt-text-missing}
3754 {
3755   Alternate~text~for~#1~is~missing.~
3756   Using~the~default~value~'#2'~instead.
3757 }

```

Sockets for tex-text boxes.

```

3758 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
3759 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
3760 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
3761 {

```

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.

```

3762 \bool_if:NTF \l__luamplib_tag_usetext_bool
3763 {
3764   \tag_mc_end_push:
3765   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
3766   \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.

```

3767   \tag_mc_begin:n{tag=text}
3768   #2
3769   \tag_mc_end:
3770   \tag_struct_end:
3771   \tag_mc_begin_pop:n{ }
3772 }
3773 {
3774   \tag_suspend:n{\luamplibtagtextboxset}
3775   #2
3776   \tag_resume:n{\luamplibtagtextboxset}
3777 }
3778 }
3779 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
3780 {
3781   \bool_lazy_and:nnTF
3782   { \l__luamplib_tag_usetext_bool }
3783   { \cs_if_free_p:c {luamplib.taggedbox.#1} }
3784   {
3785     \tag_resume:n{\mplibputtextbox}
3786     \tag_mc_end:
3787     \cs_if_exist:cTF {luamplib.taggedbox.#1}
3788     {
3789       \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
3790       #2
3791       \cs_undefine:c {luamplib.taggedbox.#1}
3792     }
3793   }

```

```

3794 \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
3795 \tag_mc_begin:n{}
3796 \int_set:Nn \l_tmpa_int {#1}
3797 \tag_mc_reset_box:N \l_tmpa_int
3798 #2
3799 \tag_mc_end:
3800 }
3801 \tag_mc_begin:n{artifact}
3802 }
3803 {
3804 \int_set:Nn \l_tmpa_int {#1}
3805 \tag_mc_reset_box:N \l_tmpa_int
3806 #2
3807 }
3808 }
3809 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
3810 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
3811 \cs_set_nopar:Npn \luamplibtagtextboxset
3812 {
3813 \tag_socket_use:nnn{luamplib/texttext/set}
3814 }

```

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

```

3815 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
3816 {
3817 \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usetext_bool
3818 \bool_set_false:N \l__luamplib_tag_usetext_bool
3819 \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}
3820 \cs_get_nopar:cpn {luamplib.notaggedbox.#1}{#1}
3821 \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
3822 }
3823 \sys_if_output_pdf:TF
3824 {
3825 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
3826 {
3827 \pdfextension save\relax
3828 \vbox to 0pt{\vss
3829 \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
3830 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
3831 \kern #3bp}
3832 \pdfextension restore\relax
3833 }
3834 }
3835 {
3836 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
3837 {
3838 \special{pdf:btrans~matrix~#4~#2~#3}
3839 \vbox to 0pt{\vss\hbox bdir0 to 0pt{

```



```

3840 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
3841 \special{pdf:etrans}
3842 }
3843 }

```

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

```

3844 \cs_set_nopar:Npn \luamplibtagasgroupset
3845 {
3846 \bool_set_false:N \l__luamplib_tag_usetext_bool
3847 }
3848 \cs_set_nopar:Npn \luamplibtagasgroupput
3849 {
3850 \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
3851 \tag_socket_use:nnn{luamplib/mpplibgroup/put}
3852 }

```

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

```

3853 \socket_new:nn{tagsupport/luamplib/mpplibgroup/put}{2}
3854 \socket_new_plug:nnn{tagsupport/luamplib/mpplibgroup/put}{default}
3855 {
3856 \cs_if_free:cT {luamplib.mpplibgroup.text.#1}
3857 {
3858 \msg_warning:nnn {luamplib} {mpplibgroup-text-mode} {#1}
3859 \cs_gset_nopar:cpn {luamplib.mpplibgroup.text.#1} {#1}
3860 }
3861 \tag_mc_end:
3862 \tag_mc_begin:n{tag=text}
3863 #2
3864 \tag_mc_end:
3865 \tag_mc_begin:n{artifact}
3866 }
3867 \socket_assign_plug:nn{tagsupport/luamplib/mpplibgroup/put}{default}

```

A macro for BBox attribute

```

3868 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
3869 {
3870 \tl_set:Ne \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
3871 \tex_savepos:D
3872 \property_record:ee{\l_tmpa_tl}{xpos,ypos}
3873 \tl_set:Ne \l__luamplib_BBox_llx_tl
3874 { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
3875 \tl_set:Ne \l__luamplib_BBox_lly_tl
3876 { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
3877 \tl_set:Ne \l__luamplib_BBox_urx_tl
3878 { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
3879 \tl_set:Ne \l__luamplib_BBox_ury_tl
3880 { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
3881 \bool_if:NT \l__luamplib_tag_bboxcorr_bool
3882 {

```

```

3883 \int_zero:N \l_tmpa_int
3884 \tl_map_inline:nn
3885 {
3886   \l__luamplib_BBox_llx_tl
3887   \l__luamplib_BBox_lly_tl
3888   \l__luamplib_BBox_urx_tl
3889   \l__luamplib_BBox_ury_tl
3890 }
3891 {
3892   \int_incr:N \l_tmpa_int
3893   \tl_set:Ne ##1
3894   {
3895     \fp_eval:n
3896     {
3897       ##1
3898       +
3899       \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
3900     }
3901   }
3902 }
3903 }
3904 \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
3905 {
3906   /O /Layout /BBox [
3907     \l__luamplib_BBox_llx_tl\c_space_tl
3908     \l__luamplib_BBox_lly_tl\c_space_tl
3909     \l__luamplib_BBox_urx_tl\c_space_tl
3910     \l__luamplib_BBox_ury_tl
3911   ]
3912 }
3913 \bool_if:NT \l__tag_graphic_debug_bool
3914 {
3915   \iow_log:e
3916   {
3917     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
3918     \l__luamplib_BBox_llx_tl\c_space_tl
3919     \l__luamplib_BBox_lly_tl\c_space_tl
3920     \l__luamplib_BBox_urx_tl\c_space_tl
3921     \l__luamplib_BBox_ury_tl
3922   }
3923   \sys_if_output_pdf:TF
3924   {
3925     \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
3926     {
3927       \pdfextension save\relax
3928       \opacity_select:n{0.5} \color_select:n{red}
3929       \pdfextension literal~text
3930       {
3931         \l__luamplib_BBox_llx_tl\c_space_tl

```

```

3932     \l__luamplib_BBox_lly_tl\c_space_tl
3933     \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3934     \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3935     re~f
3936   }
3937   \pdfextension restore\relax
3938 }
3939 }
3940 {
3941   \tl_set:Nc \l__luamplib_tag_bbox_draw_tl
3942   {
3943     \special{pdf:bcontent}
3944     \opacity_select:n{0.5} \color_select:n{red}
3945     \special{pdf:code~
3946       1~0~0~1~
3947       -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
3948       -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
3949       cm
3950     }
3951     \special{pdf:code~
3952       \l__luamplib_BBox_llx_tl\c_space_tl
3953       \l__luamplib_BBox_lly_tl\c_space_tl
3954       \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
3955       \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
3956       re~f
3957     }
3958     \special{pdf:econtent}
3959   }
3960 }
3961 }
3962 }

```

Sockets for main process

```

3963 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
3964 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
3965 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
3966 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
3967 {
3968   \tag_mc_end_push:
3969   \tl_if_empty:NT\l__luamplib_tag_alt_tl
3970   {
3971     \tl_if_empty:eTF{#1}
3972     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
3973     { \tl_set:Nc \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
3974     \msg_warning:nnVV{luamplib}{alt-text-missing}
3975     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
3976   }
3977   \tag_struct_begin:n
3978   {

```

```

3979     tag=\l__luamplib_tag_struct_tl,
3980     alt=\l__luamplib_tag_alt_tl,
3981   }
3982   \tag_mc_begin:n{}
3983 }
3984 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
3985 {
3986   \__luamplib_tag_bbox_attribute:n {#1}
3987   #2
3988   \tl_use:N \l__luamplib_tag_bbox_draw_tl
3989   \tag_mc_end:
3990   \tag_struct_end:
3991   \tag_mc_begin_pop:n{}
3992 }
3993 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
3994 {
3995   \tag_mc_end_push:
3996   \tag_struct_begin:n
3997   {
3998     tag=Span,
3999     actualtext=\l__luamplib_tag_actual_tl,
4000   }
4001   \tag_mc_begin:n{}
4002 }
4003 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4004 {
4005   #2
4006   \tag_mc_end:
4007   \tag_struct_end:
4008   \tag_mc_begin_pop:n{}
4009 }
4010 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4011 {
4012   \tag_mc_end_push:
4013   \tag_mc_begin:n{artifact}
4014 }
4015 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4016 {
4017   #2
4018   \tag_mc_end:
4019   \tag_mc_begin_pop:n{}
4020 }

```

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

```

4021 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4022 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4023 {
4024   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}

```

```

4025 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4026 }
4027 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4028 {
4029 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4030 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4031 \prependtomplibbox \mplibnoforcehmode
4032 \mode_if_vertical:T { \noindent \aftergroup\par }
4033 }
4034 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4035 {
4036 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4037 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4038 }
4039 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4040 {
4041 \bool_set_true:N \l__luamplib_tag_usetext_bool
4042 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4043 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4044 \prependtomplibbox \mplibnoforcehmode
4045 \mode_if_vertical:T { \noindent \aftergroup\par }
4046 }
4047 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4048 {
4049 \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4050 \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4051 }
4052 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```

4053 \keys_define:nn{luamplib/tagging}
4054 {
4055 ,alt .code:n =
4056 {
4057 \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4058 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4059 }
4060 ,actualtext .code:n =
4061 {
4062 \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4063 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4064 }
4065 ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4066 ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4067 ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4068 ,tag .code:n =
4069 {
4070 \str_case:nnF {#1}

```

```

4071 {
4072   {false} { \keys_set:nn {luamplib/tagging} {off} }
4073   {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4074 }
4075 {
4076   \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4077   \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4078 }
4079 }
4080 ,adjust-BBox .code:n =
4081 {
4082   \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4083   \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4084 }
4085 ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4086 }
4087 \keys_define:nn {luamplib/instance}
4088 {
4089   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4090   ,instancename .meta:n = { instance = {#1} }
4091   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4092 }

```

Redefine our macros

```

4093 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4094 {
4095   \prependtomplibbox
4096   \hbox dir~TLT\bgroup
4097     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4098     \xdef\MPllx{#1}\xdef\MPlly{#2}%
4099     \xdef\MPurx{#3}\xdef\MPury{#4}%
4100     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4101     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4102     \parskip0pt
4103     \leftskip0pt
4104     \parindent0pt
4105     \everypar{}%
4106     \setbox\mplibscratchbox\vbox\bgroup
4107       \tag_suspend:n{\mplibstarttoPDF}
4108       \noindent
4109 }
4110 \cs_set_nopar:Npn \mplibstoptoPDF
4111 {
4112   \par
4113   \egroup
4114   \setbox\mplibscratchbox\hbox
4115     {\hskip-\MPllx bp
4116     \raise-\MPlly bp
4117     \box\mplibscratchbox}%

```

```

4118 \setbox\mplibscratchbox\ vbox to \MPheight
4119 {\vfill
4120 \hsize\MPwidth
4121 \wd\mplibscratchbox\0pt
4122 \ht\mplibscratchbox\0pt
4123 \dp\mplibscratchbox\0pt
4124 \box\mplibscratchbox}%
4125 \wd\mplibscratchbox\MPwidth
4126 \ht\mplibscratchbox\MPheight
4127 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4128 \egroup
4129 }
4130 \RenewDocumentCommand\mplibcode{0{}}
4131 {
4132 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4133 \tl_gclear:N \currentmpinstancename
4134 \keys_set:known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4135 \keys_set:nV {luamplib/instance} \l_tmpa_tl
4136 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4137 \tag_socket_use:n{luamplib/figure/init}
4138 \mplibtmptoks{}\ltxdomplibcode
4139 }
4140 \RenewDocumentCommand\mpfig{s 0{}}
4141 {
4142 \begingroup
4143 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4144 \keys_set:known:ne {luamplib/tagging} {#2}
4145 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4146 \tag_socket_use:n{luamplib/figure/init}
4147 \IfBooleanTF{#1} { \mplibprempfig * }
4148 { \mplibmainmpfig }
4149 }
4150 \RenewDocumentCommand\usemplibgroup{0{ } m}
4151 {
4152 \begingroup
4153 \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4154 \keys_set:known:ne {luamplib/tagging} {#1}
4155 \tag_socket_use:n{luamplib/figure/init}
4156 \prependtomplibbox\hbox dir~TLT\bgroup
4157 \tag_socket_use:nn{luamplib/figure/begin}{#2}
4158 \setbox\mplibscratchbox\hbox\bgroup
4159 \bool_if:NF \l__luamplib_tag_usetext_bool { \tag_suspend:n{\usemplibgroup} }
4160 \tag_socket_use:nnn{luamplib/mpfiggroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4161 \egroup
4162 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4163 \endgroup
4164 \endgroup
4165 }

```

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

well.

```
4166 \cs_new_nopar:Npn \mplibalttext #1
4167 {
4168   \tl_set:Nc \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4169 }
4170 \cs_new_nopar:Npn \mplibactualtext #1
4171 {
4172   \tl_set:Nc \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4173 }
4174 \ExplSyntaxOff
```

That's all folks!

3 The GNU GPL License v2

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

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

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

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

Preamble

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

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

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

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

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

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NO WARRANTY

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

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

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

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

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

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

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

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

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

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

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

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

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

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

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

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subcomponent library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.