

The luamplib package

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

Current Maintainer: Kim Dohyun

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

2026/07/09 v2.42.3

Abstract

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

Contents

1	Documentation	2
1.1	\TeX	3
1.1.1	Forcing horizontal mode	3
1.1.2	Insert some code into every code chunk	3
1.1.3	Choosing a METAPOST format	3
1.1.4	Choosing a number system	4
1.1.5	Showing METAPOST log	4
1.1.6	verbatimtex Legacy behavior	5
1.1.7	\TeX -text labels	5
1.1.8	Inherit METAPOST code	6
1.1.9	<code>\mplibglobaltexttext</code>	6
1.1.10	Separate METAPOST instances	6
1.1.11	Verbatim input	7
1.1.12	\TeX dimen	7
1.1.13	\TeX color	7
1.1.14	<code>\mpfig</code> , <code>\endmpfig</code>	8
1.1.15	About cache files	9
1.1.16	About figure box metric	9
1.1.17	<code>luamplib.cfg</code>	9
1.1.18	Tagged PDF	9
1.2	METAPOST	11
1.2.1	\TeX dimen and color	11
1.2.2	More on \TeX color	11
1.2.3	Fill and stroke colors	12
1.2.4	Transparency	12

1.2.5	Fill and stroke transparency	13
1.2.6	Text outline as a text image	13
1.2.7	Glyph outline	14
1.2.8	Drawing a glyph outline	14
1.2.9	Text outline as a path image	15
1.2.10	Unicode-aware length	15
1.2.11	Unicode-aware substring	15
1.2.12	Shading	16
1.2.13	Fading	18
1.2.14	Tiling pattern	19
1.2.15	Form XObject from METAPOST side	21
1.2.16	Form XObject from T _E X side	24
1.2.17	Masking	25
1.2.18	Free-form Gouraud-shaded triangle mesh shading	26
1.2.19	Lattice-form Gouraud-shaded triangle mesh shading	28
1.2.20	Coons patch mesh shading	28
1.2.21	Tensor-product patch mesh shading	30
1.3	Lua	31
1.3.1	runscript	31
1.3.2	Accessing METAPOST variables	32
1.3.3	Running a METAPOST code	32
1.3.4	Registering a tiling pattern	33
1.3.5	Registering a form XObject	33
2	Implementation	34
2.1	Lua module	34
2.2	T _E Xpackage	114
3	The GNU GPL License v2	135

1 Documentation

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

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

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

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

- Possibility to use `btex ... etex` to typeset \TeX code. `texttext <string>` is a more versatile macro equivalent to `TEX <string>` from `TEX.mp`. `TEX` is also allowed and is a synonym of `texttext`. The argument of `mplib`'s primitive `maketext` will also be processed by the same routine.
- Possibility to use `verbatimtex ... etex` to run a \TeX code. `VerbatimTeX <string>` is a more versatile macro corresponding to `verbatimtex` command. Of course the behavior cannot be the same as the stand-alone `mpost`, so that you cannot include `\documentclass`, `\usepackage` etc. When these \TeX commands are found in `verbatimtex ... etex`, the entire code will be ignored.

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

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

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

1.1 \TeX

1.1.1 Forcing horizontal mode

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

1.1.2 Insert some code into every code chunk

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

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



1.1.3 Choosing a `METAPOST` format

There are (basically) two formats for `METAPOST`: *plain* and *metafun*. By default, the *plain* format is used, but you can set the format to be used by future figures at any time using `\mplibsetformat <format name>`.

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.

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

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

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

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

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

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

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

where *<string>* should be "" (empty), "isolated", "knockout", or "isolated, knockout". Beware that currently many of the PDF rendering applications, except Adobe Acrobat and T_EXworks, cannot properly render the isolated or knockout effect.

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

1.1.4 Choosing a number system

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

1.1.5 Showing METAPOST log

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

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

1.1.6 verbatimtex Legacy behavior


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

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

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

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

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



diameter: 22.62764bp.

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

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

ABC **DEF GHI**

1.1.7 \TeX -text labels

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

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

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

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

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

1.1.8 Inherit `METAPOST` code

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

1.1.9 `\mplibglobaltexttext{enable|disable}`

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

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



1.1.10 Separate `METAPOST` instances

`luamplib` v2.22 has added the support for several named `METAPOST` instances in \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 `\mplibglobaltext`.
- When an instance name 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 Verbatim input

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

1.1.12 \TeX `dimen`

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

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



1.1.13 \TeX color

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

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

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

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



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

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

1.1.14 `\mpfig ... \endmpfig`

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

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

and the starred version is roughly the same as follows:

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

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

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



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

1.1.15 About cache files

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

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

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

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

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

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

1.1.16 About figure box metric

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

1.1.17 `luamplib.cfg`

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

1.1.18 Tagged PDF

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

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

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

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

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

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

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

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

```

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

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

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

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

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

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

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

```

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

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

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

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

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

```

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

1.2 METAPOST

1.2.1 T_EX dimen and color

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

1.2.2 More on T_EX color

`mplibtexcolor \langle string \rangle` is a METAPOST operator that converts a T_EX color expression to a METAPOST color expression, that can be used anywhere color expression is expected as well as after

the `withcolor` command.⁹ For instance:

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

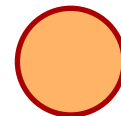
But the result may vary in its color model (gray/rgb/cmyk) according to the given \TeX color. Therefore the example shown above would raise a `METAPOST` error: `cmykcolor col; should have been declared`. By contrast, `mplibrgbtexcolor` $\langle string \rangle$ always returns rgb-model expressions.

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

1.2.3 Fill and stroke colors

Unlike the `withcolor` command, users can specify one color for filling and another color for stroking using the macro `withmplibcolors` at the end of a sentence. The syntax is `withmplibcolors` $(\langle fill\ color\ expr \rangle, \langle stroke\ color\ expr \rangle)$. When the argument is in string type, it is regarded as the color expression of \TeX side. A simple example (see also the example at § 1.2.8):

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

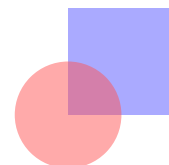


The PDF file size is much smaller than issuing two sentences with different colors, though the apparent effect is the same.

1.2.4 Transparency

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

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

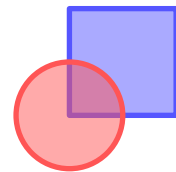


⁹Since v2.38.1, the operation of `mplibtexcolor` is the same as that of `mplibcolor` if the color specified is not a spot color.

1.2.5 Fill and stroke transparency

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.

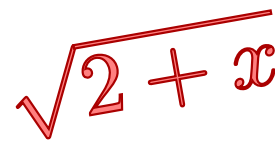
```
\mpfig
pickup pencircle scaled 2;
filldraw unitsquare scaled 40
  withcolor 1/3[blue,white]
  withmplibopacities (1, 1/2, 1)    % or ("normal", 1/2, 1)
;
filldraw fullcircle scaled 40
  withcolor 1/3[red,white]
  withmplibopacities (1, 1/2, 1)
;
\endmpfig
```



1.2.6 Text outline as a text image

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

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



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

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

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

¹¹But this limitation is now lifted by the introduction of `withshadingmethod`. See below § 1.2.12.

1.2.7 Glyph outline

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

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

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

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

1.2.8 Drawing a glyph outline

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

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

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

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

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

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

An example:

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



1.2.9 Text outline as a path image

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

A simple example:

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



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

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

1.2.10 Unicode-aware length

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

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

1.2.11 Unicode-aware substring

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

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

1.2.12 Shading

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

- *Textual pictures* as well as paths can have shading effect. The term *textual picture* here means a picture generated by `btex ... etex`, `texttext`, `TEX`, `maketext`, `mplibgraphicstext` (see below § 1.2.6), or `infont` operator, though technically only the last one is a true textual picture. Note that the picture, including transparency group, in which the objects are filled *without* color can also be regarded as a textual picture.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 to the *metafun*'s: `withshadingpoints`, `withshadingcenters`, `withshadingextend`, `withshadingmatrix`, and `withshadingstroke`.
- More shading methods are available: "triangle", "lattice", "coons", and "tensor". See below § 1.2.18, § 1.2.19, § 1.2.20 and § 1.2.21 for these methods.

The syntax is $\langle path \rangle | \langle textual picture \rangle$ `withshadingmethod` $\langle string \rangle$, where the latter shall be "linear", "circular", "triangle", "lattice", "coons", or "tensor". The balance of this subsection is to explain additional optional macros.

Above all, there are two ways in specifying the shading coordinates, of which you can choose the more convenient one. First, the way that mimicks the *metafun*'s:

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

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

¹⁴See below § 1.2.8, particularly the first *example* of tiling pattern at § 1.2.14. See also § 1.2.15 and § 1.2.16, particularly the *example* in the *note* about picture shading and transparency group.

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

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

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

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

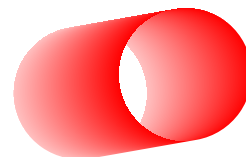
withshadingextend ($\langle boolean \rangle, \langle boolean \rangle$) Values specifying whether to extend the shading beyond the starting and ending points or circles, default value being (true, true). An example just to show the concept:

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

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

1.2.13 Fading

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

Related macros to control optional values are:

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

withfadevector ($\langle pair \rangle$, $\langle pair \rangle$) 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 ($\langle numeric \rangle$, $\langle numeric \rangle$) 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 $\langle numeric \rangle$ Fractional number of each fading step. Only meaningful within `withfadestep`.

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

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

withfadebbox ($\langle pair \rangle$, $\langle pair \rangle$) sets the bounding box of the fading area, default value being (llcorner p, urcorner p). Though this option is not needed in most cases, there could be cases when users want to explicitly control the bounding box. Particularly, see the description [below](#) at § 1.2.15 on the analogous macro withgroupbbox.

An example:

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



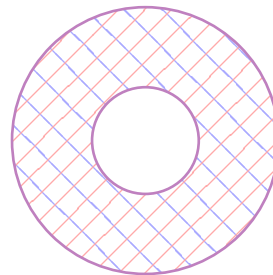
1.2.14 Tiling pattern

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

An example:

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

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



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

Table 1: options for `\mppattern`

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

* in string type, numbers are separated by spaces

```

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

```

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

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

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

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

```

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

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

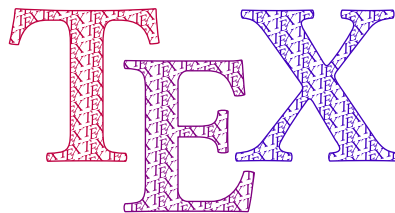
```

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

```

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

```



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

```

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

```



1.2.15 Form XObject from METAPost side

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

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

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

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

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

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

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

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

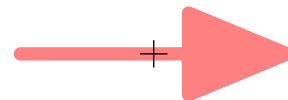
withgroupbbox ($\langle pair \rangle$, $\langle pair \rangle$) sets the bounding box of the XObject, default value being (llcorner p , urcorner p). This option might be needed especially when you draw with a thick pen a path that touches the boundary; you would probably want to append to the sentence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

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

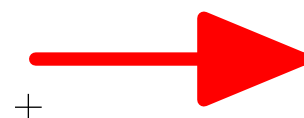
```
\mpfig
picture pic;
pic = image(drawarrow (left--right) scaled 5 withcolor red) scaled 10 ;
draw pic
  asgroup "off"
  withtransparency (1, 1/2) ;
\endmpfig
```



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



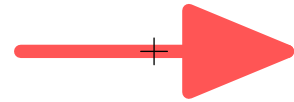
```
\noindent
\clap{\vrule width 10bp height .25bp depth .25bp}%
\clap{\vrule width .5bp height 5bp depth 5bp}%
\usemplibgroup{mygroup}
```



```

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

```



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

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

```

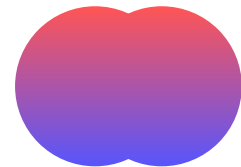
\mpfig*
  picture pic[];
  pic1 = image(
    fill fullcircle scaled 60 withoutcolor;
    fill fullcircle scaled 60 shifted 25right withoutcolor;
  );
  pic2 = image(
    draw pic1
    withshadingmethod "linear"
    withshadingvector (2, 1)
    withshadingcolors (red, blue)
  );
\endmpfig

```

```

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

```



```

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

```



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

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

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



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

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

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



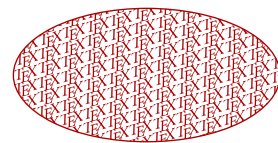
1.2.16 Form XObject from TeX side

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

An example:¹⁷

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

\usemplibgroup{texpatt}
```



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

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

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

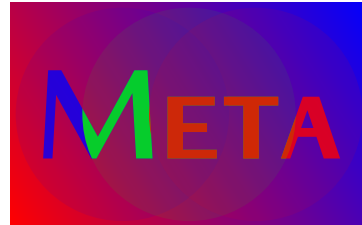
withmaskingbgcolor \langle *color expr* \rangle sets the background color of the masking group. 0 denotes full transparency (masking invisibly); 1, full color.²⁰

An example:

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

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

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



1.2.18 Free-form Gouraud-shaded triangle mesh shading

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

Optional macros for triangle mesh shading:

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

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

withtrianglepatchinit ($\langle path \rangle$ | $\langle string \rangle$, $\langle color \rangle$, $\langle color \rangle$, $\langle color \rangle$) The initial triangle. This macro can be used multiple times.

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

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

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

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

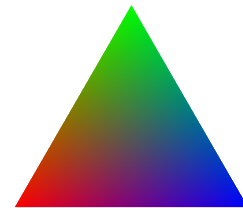
For specifying a new vertex which determines the next triangle, the second argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates).

The third argument is the color for the new vertex.

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

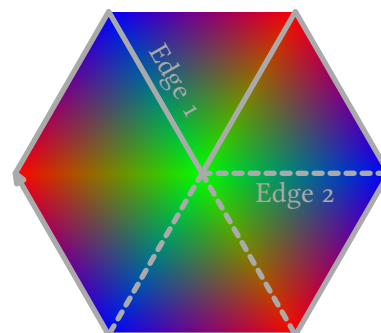
```
\mpfig
pair q ;
vardef qadd (expr a) =
  q := q shifted (dir a * 70) ;
  q
enddef;

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



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

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



1.2.19 Lattice-form Gouraud-shaded triangle mesh shading

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

Optional macros for lattice-form shading:

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

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

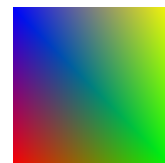
The first argument shall be a pair, or a string of two numerics separated by a space (x and y coordinates). The second argument shall be a color expression for the vertex.

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

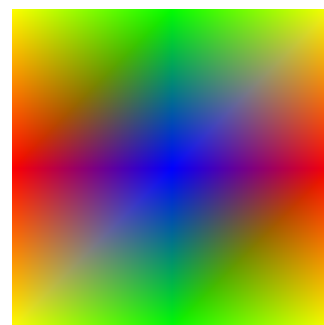
When this macro is not given, the default values will be four points of the path obtained from the object of shading and red, green, yellow, and blue for the colors at each point.

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

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



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



1.2.20 Coons patch mesh shading

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

Optional macros for Coons patch mesh shading:

withcoonspatchinit ($\langle path \rangle$ | $\langle string \rangle$, $\langle color \rangle$, $\langle color \rangle$, $\langle color \rangle$, $\langle color \rangle$) The initial patch. This macro can be used multiple times.

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

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

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

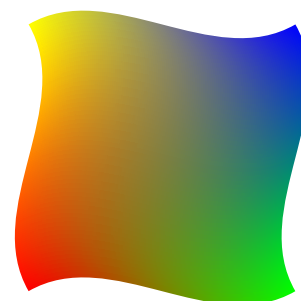
The first argument shall be a number (1, 2, or 3) denoting the previous patch's edge a new patch will be attached to. For instance, edge 1 is the segment between point 1 and point 2 of the previous patch.

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

Remaining arguments are two color expressions for the new vertices.

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

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

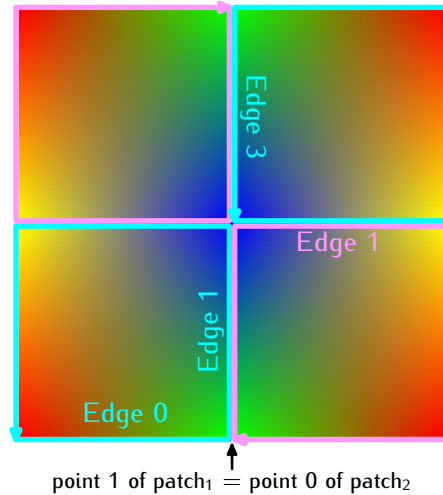


```
\mpfig
u := 80 ;
path p;
p = unitsquare scaled u;
def fsquare (expr n) =
  ( point n of p --
    point n+1 of p --
    point n+2 of p --
    point n+3 of p --
    cycle )
enddef;
```

```

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

```

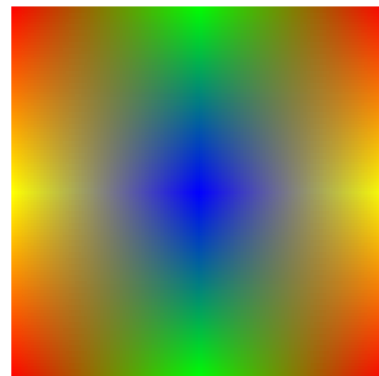


N.B. Be aware that currently Mac OS Preview exposes its some bug in rendering the figure just above, especially when the edge flag is 3. In order to circumvent the Preview's bug, you can use withcoonspatchinit multiple times:

```

\mpfig
u := 70 ;
p := unitsquare scaled u ;
fill p scaled 2
  withshadingmethod "coons"
  withcoonspatchinit
    (p, red, green, blue, yellow)
  withcoonspatchinit
    (p shifted (u,0), green, red, yellow, blue)
  withcoonspatchinit
    (p shifted (u,u), blue, yellow, red, green)
  withcoonspatchinit
    (p shifted (0,u), yellow, blue, green, red) ;
\endmpfig

```



1.2.21 Tensor-product patch mesh shading

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

Optional macros for tensor-product patch mesh shading:

withtensorpatchinit ($\langle path \rangle | \langle string \rangle, \langle path \rangle | \langle string \rangle, \langle color \rangle, \langle color \rangle, \langle color \rangle, \langle color \rangle$) The initial patch. This macro can be used multiple times.

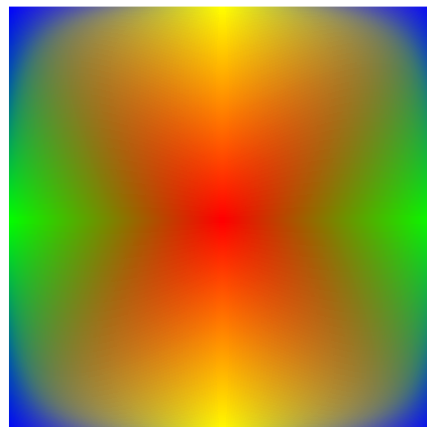
The only difference to `withcoonspatchinit` macro of Coons patch shading is the second argument inserted for specifying four inner control points. It shall be a path that has four points, or a string composed of eight numerics separated by spaces (x and y coordinates of each point). If the first argument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

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

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

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

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



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

1.3 Lua

1.3.1 `runscript` ...

A good many METAPOST macros described in this documentation have been implemented using the primitive `runscript`. With `runscript` $\langle string \rangle$, 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 Accessing METAPOST variables

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

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

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

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

```
\def\mpnumeric#1#2{\directlua{
  tex.sprint(tostring(luamplib.instances["#1"]:get_numeric"#2"))
}}
\mpnumeric{myinstance}{n}\relax 3.0
```

1.3.3 Running a METAPOST code

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

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

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

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

Table 3: elements in luamplib table (partial)

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

1.3.4 Registering a tiling pattern

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

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

The first argument is the register of a box containing a pattern cell, which should be prepared in advance by the user. For instance, `\setbox0=\hbox{\tiny\TeX}`, or corresponding Lua code using `tex.setbox` function; then the argument should be \emptyset .²¹

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

1.3.5 Registering a form XObject

This is the Lua interface for `\mplibgroup ... \endmplibgroup` described above at § 1.2.16.

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

The first argument is the register of a box prepared in advance by the user. When the contents of the box have been generated from \TeX (not `METAPOST`) code, please make sure that both of the \TeX macros ‘`MP11x`’ and ‘`MP11y`’ are defined as ‘ \emptyset ’ before invoking the Lua function.²²

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

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

²¹In DVI mode, \TeX 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.

²²In DVI mode, \TeX macro ‘`mplibgroupname`’ also should be set as $\langle group name \rangle$ before preparing the box, if shading pattern (i.e. shading on picture) is used in the `mplibgroup`.

2 Implementation

2.1 Lua module

```
1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.42.3",
5   date      = "2026/07/09",
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+")
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 texsprintf = tex.sprintf
54 local texgettoks = tex.gettoks
55 local texgetbox = tex.getbox
56 local texruntoks = tex.runtoks
57 if not texruntoks then
58 err("Your LuaTeX version is too old. Please upgrade it to the latest")
59 end
60 local is_defined = token.is_defined
61 local get_macro = token.get_macro
62 local mplib = require ('mplib')
63 local kpse = require ('kpse')
64 local lfs = require ('lfs')
65 local lfsattributes = lfs.attributes
66 local lfsisdir = lfs.isdir
67 local lfsmkdir = lfs.mkdir
68 local lfstouch = lfs.touch
69 local iopen = io.open
70

```

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

```

71 local file = file or { }
72 local replacesuffix = file.replacesuffix or function(filename, suffix)
73 return (filename:gsub("%.[%a%d]+$", "")) .. "." .. suffix
74 end
75 local is_writable = file.is_writable or function(name)
76 if lfsisdir(name) then
77 name = name .. "/_luam_plib_temp_file_"
78 local fh = iopen(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   lfsmkdir(full)
90   end
91 end
92

```

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

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

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

129 local noneedtoreplace = {
130   ["boxes.mp"] = true, -- ["format.mp"] = true,
131   ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,

```

```

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

```

Pattern formats to replace bte_x and verbatim_{tex} . . . et_{ex} in input files, if needed.

```

147 local name_b = "%f[%a_]"
148 local name_e = "%f[^%a_]"
149 local bteex_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 LuaTeX manual, which is not the case of standalone METAPOST though.

```

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

```

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

```

208 local mpkpse
209 do
210   local exe = 0
211   while arg[exe-1] do
212     exe = exe-1
213   end
214   mpkpse = kpse.new(arg[exe], "mpost")
215 end
216 local special_ftype = {
217   pfb = "type1 fonts",
218   enc = "enc files",
219 }
220 function luamplib.finder (name, mode, ftype)

```

```

221   if mode == "w" then
222     if name and name ~= "mpout.log" then
223       kpse.record_output_file(name) -- recorder
224     end
225     return name
226   else
227     ftype = special_ftype[ftype] or ftype
228     local file = mpkpse:find_file(name,ftype)
229     if file then
230       if lfstouch and ftype == "mp" and not noneedtoreplace[name] and not noneedtoreplace["*.mp"] then
231         file = replaceinputmpfile(name,file)
232       end
233     else
234       file = mpkpse:find_file(name, name:match("%a+$"))
235     end
236     if file then
237       kpse.record_input_file(file) -- recorder
238     end
239     return file
240   end
241 end
242 end
243

```

For the main function: process

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

```

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

```

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

```

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

```

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

```

260     local log = l or t or "no-term"
261     log = log:gsub("%(Please type a command or say `end'%)", ""):gsub("\n+", "\n")
262     if result.status > 0 then
263       local first = log:match"(.-\n! .-)\n! "
264       if first then

```

```

265     termorlog("term", first)
266     termorlog("log", log, "Warning")
267   else
268     warn(log)
269   end
270   if result.status > 1 then
271     err(e or "see above messages")
272   end
273   elseif prevlog then
274     log = prevlog..log

```

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

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

```

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

```

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

```

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

```

Make use of make_text and run_script. And we provide numbersystem option since v2.4. See

<https://github.com/lualatex/luamplib/issues/21>.

```

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

```

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

```

299 local preamble = tableconcat{
300   format(luamplib.preambles.preamble, replacesuffix(name,"mp")),
301   luamplib.preambles.mplibcode,
302   luamplib.legacyverbatim and luamplib.preambles.legacyverbatim or "",
303   luamplib.texttextlabel and luamplib.preambles.texttextlabel or "",
304 }

```

```

305 local result, log
306 if not mpx then
307   result = { status = 99, error = "out of memory"}
308 else
309   result = mpx:execute(preamble)
310 end
311 log = reporterror(result)
312 return mpx, result, log
313 end

```

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

```

314 local process_stack = 0
315 function process (data, instancename)
316   local currfmt
317   process_stack = process_stack + 1
318   if instancename and instancename ~= "" then
319     currfmt = instancename
320     has_instancename = true
321   else
322     currfmt = tableconcat{
323       currentformat,
324       luamplib.numbersystem or "scaled",
325       tostring(luamplib.texttextlabel),
326       tostring(luamplib.legacyverbatimimtex),
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 LuaTeX'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() textsprint(cat or catlatex, str) end)
364 end

```

For conversion of sp to bp.

```

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

```

Prepare text 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 textext_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.globaltextext 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 is_xcolor (str)
421   for _,v in ipairs(str:explode"!") do
422     if not v:find("^%s*d+%s*$") and is_defined("\\color@".v) then -- priority to xcolor
423       return true
424     end
425   end
426   return false
427 end
428 local function colorsplit (res)
429   local t, tt = { }, res:gsub("[%%]", "", 2):explode()
430   local be = tt[1]:find"^d" and 1 or 2
431   for i=be, #tt do
432     if not tonumber(tt[i]) then break end
433     t[#t+1] = tt[i]
434   end

```

```

435 return t
436 end
437 do
438 local colfmt = ccexplat and "l3color" or "xcolor"
439 local mplibcolorfmt = {
440   xcolor = [[{\setbox0\hbox{{\color%sglobal\mplibtmptoks\expandafter{\current@color}}}}]],
441   l3color = is_defined "__color_select:nn" and tableconcat{ -- to be cleaned up
442     [[\begingroup\def__color_select:N#1{\expandafter__color_select:nn#1}]],
443     [[\def__color_backend_select:nn#1#2{\global\mplibtmptoks{#1 #2}}]],
444     [[\def__kernel_backend_literal:e#1{\global\mplibtmptoks\expandafter{\expanded{#1}}}],
445     [[\color_select:n%svendgroup]],
446   } or is_defined "__color_export_format_raw:nnN" and
447     [[\color_export:nnN%sv{raw}\l_tmpa_tl\mplibtmptoks\expandafter{\l_tmpa_tl}]]
448   or [[{\setbox0\hbox{{\color_select:n%sglobal\mplibtmptoks\expanded{\current@color}}}}]]
449 }
450 function process_color (str)
451   if str then
452     if not str:find("%b{") then
453       str = format("%s",str)
454     end
455     local myfmt = mplibcolorfmt[colfmt]
456     if colfmt == "l3color" and is_defined "color" then
457       if str:find("%b[") or is_xcolor(str:match("{(.+)")) then
458         myfmt = mplibcolorfmt.xcolor
459       end
460     end
461     run_tex_code(myfmt:format(str), ccexplat or catat11)
462     local t = texgettoks"mplibtmptoks"
463     if not pdfmode then
464       if t:find" cs " then -- spot color raw export
465         local name = t:match("^/(.-) cs ")
466         texsprint(ccexplat, {
467           "\\pdfmanagement_add:nnn{Page/Resources/ColorSpace}{",
468             name, "}{\\pdf_object_ref:n{", name, "}"
469         })
470       elseif t:find"^hsb" then
471         local spec = t:gsub("^hsb ", ""):gsub(" ", ",")
472         run_tex_code{"\\convertcolorspec{hsb}{", spec, "}{rgb}\\mplibtmpa"}
473         t = format("rgb %s", get_macro"mplibtmpa":gsub(","," "))
474       elseif not t:find"%d" then -- named color
475         run_tex_code{"\\extractcolorspecs{", t, "}\\mplibtmpa\\mplibtmpb"}
476         t = format("%s %s", get_macro"mplibtmpa", get_macro"mplibtmpb":gsub(","," "))
477       end
478     elseif is_defined"ver@colorspace.sty" and t:find"/&" then
479       local a,b,c,d = t:match"^(.- cs) (.- CS) (.- scn?) (.- SCN?)$"
480       if a and b and c and d then
481         t = tableconcat({ a, c, b, d }, " ")
482       end
483     end

```

```

484     return format('1 withprescript "mpliboverridecolor=%s"', t)
485   end
486   return ""
487 end
488 function process_mplibcolor(str)
489   local res = process_color(str)
490   if res:find" cs " or res:find"@pdf.obj" then return res end
491   res = colorsplit(res:match"mpliboverridecolor=(.+)")
492   return format("%s", tableconcat(res, ", "))
493 end
494 end
495
   for \mpdim or mplibdimen
496 local function process_dimen (str)
497   if str then
498     str = str:gsub("{(.+)}", "%1")
499     run_tex_code(format([[ \mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
500     return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
501   end
502   return ""
503 end
504

```

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

```

505 local function process_verbatimtex_text (str)
506   if str then
507     run_tex_code(str)
508   end
509   return ""
510 end
511

```

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.

```

512 local tex_code_pre_mplib = {}
513 luamplib.figid = 1
514 luamplib.in_the_fig = false
515 local function process_verbatimtex_prefig (str)
516   if str then
517     tex_code_pre_mplib[luamplib.figid] = str
518   end
519   return ""
520 end
521 local function process_verbatimtex_infig (str)
522   if str then
523     return format('special "postmplibverbtex=%s";', str)
524   end
525   return ""
526 end

```

527

For *metafun* format. see issue #79.

```
528 mp = mp or {}
529 local mp = mp
530 mp.mf_path_reset = mp.mf_path_reset or function() end
531 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
532 mp.report = mp.report or info
```

metafun 2021-03-09 changes crashes luamplib.

```
533 catcodes = catcodes or {}
534 local catcodes = catcodes
535 catcodes.numbers = catcodes.numbers or {}
536 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
537 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
538 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
539 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
540 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
541 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
542 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
543
```

Now luamplib.runscript

```
544 do
545   local runscript_funcs = {
546     luamplibtext    = process_tex_text,
547     luamplibcolor   = process_mplibcolor,
548     luamplibdimen   = process_dimen,
549     luamplibprefig  = process_verbatimtex_prefig,
550     luamplibinfig   = process_verbatimtex_infig,
551     luamplibverbtex = process_verbatimtex_text,
552   }
553
```

A function from ConT_EXt general.

```
553 local function mpprint(buffer,...)
554   for i=1,select("#",...) do
555     local value = select(i,...)
556     if value ~= nil then
557       local t = type(value)
558       if t == "number" then
559         buffer[#buffer+1] = format("%.16f",value)
560       elseif t == "string" then
561         buffer[#buffer+1] = value
562       elseif t == "table" then
563         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
564       else -- boolean or whatever
565         buffer[#buffer+1] = tostring(value)
566       end
567     end
568   end
569 end
570 function luamplib.runscript (code)
```

```

571 local id, str = code:match("(.-){(.*)}")
572 if id and str then
573     local f = runscript_funcs[id]
574     if f then
575         local t = f(str)
576         if t then return t end
577     end
578 end
579 local f = loadstring(code)
580 if type(f) == "function" then
581     local buffer = {}
582     function mp.print(...)
583         mpprint(buffer,...)
584     end
585     local res = {f()}
586     buffer = tableconcat(buffer)
587     if buffer and buffer ~= "" then
588         return buffer
589     end
590     buffer = {}
591     mpprint(buffer, tableunpack(res))
592     return tableconcat(buffer)
593 end
594 return ""
595 end
596 end
597

```

luamplib.maketext

```

598 luamplib.legacyverbatimimtex = true
599 do

```

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

```

600 local function protecttexcontents (str)
601     return str:gsub("\\%", "\\0PerCent\0")
602         :gsub("%%.\n", "")
603         :gsub("%%.-$", "")
604         :gsub("%zPerCent%z", "\\%")
605         :gsub("\r.-$", "")
606         :gsub("%s+", " ")
607 end
608 function luamplib.maketext (str, what)
609     if str and str ~= "" then
610         str = protecttexcontents(str)
611         if what == 1 then
612             if not str:find("\\documentclass"..name_e) and
613                 not str:find("\\begin%s*{document}") and
614                 not str:find("\\documentstyle"..name_e) and
615                 not str:find("\\usepackage"..name_e) then
616                 if luamplib.legacyverbatimimtex then

```

```

617         if luamplib.in_the_fig then
618             return process_verbatimtex_infig(str)
619         else
620             return process_verbatimtex_prefig(str)
621         end
622     else
623         return process_verbatimtex_text(str)
624     end
625 end
626 else
627     return process_tex_text(str, true) -- bool is for 'char13'
628 end
629 end
630 return ""
631 end
632 end
633

```

luamplib's METAPOST color operators

```

634 local function get_spc_name_obj (spot)
635   if is_defined"spc@csall" then
636     for v in get_macro"spc@csall":gmatch"{(-)}" do
637       local n, r = get_macro("spc@ir@".v):match"^(.-) (%d+ 0 R)"
638       if spot == n then
639         return v, r
640       end
641     end
642   else
643     err"only l3color or colorspace is supported for spot color"
644   end
645 end
646 do
647   local function cmyk2rgb (t)
648     return {
649       1 - math.min(1, t[1]+t[4]),
650       1 - math.min(1, t[2]+t[4]),
651       1 - math.min(1, t[3]+t[4]),
652     }
653   end
654   luamplib.gettexcolor = function (str, rgb)
655     local res = process_color(str):match"mpliboverridecolor=(.+)""
656     if res:find" cs " or res:find"@pdf.obj" then
657       if not rgb then
658         warn"%s is a spot color. Forced to CMYK", str)
659       end
660       if not is_xcolor(str) then
661         run_tex_code({
662           "\\color_export:nnN{" ,
663           str,
664           "}" ,

```

```

665     rgb and "space-sep-rgb" or "space-sep-cmyk",
666     "}\mplib@tempa",
667   },ccexplat)
668   return get_macro"mplib@tempa":explode()
669   else
670     local name, value = res:match"^(.-) cs (.-) sc"
671     local t = get_macro("spc@ascmyk@"..get_spc_name_obj(name)):explode", " -- mixed not work
672     for i = 1, #t do
673       t[i] = t[i] * value
674     end
675     return rgb and cmyk2rgb(t) or t
676   end
677 end
678 local t = colorsplit(res)
679 if #t == 3 or not rgb then return t end
680 if #t == 4 then return cmyk2rgb(t) end
681 return { t[1], t[1], t[1] }
682 end
683 end
684 luamplib.shadecolor = function (str)
685   local res = process_color(str):match"mpliboverridecolor=(.+)""
686   if res:find" cs " or res:find"@pdf.obj" then -- spot color shade

```

An example of spot color shading:

```

\DocumentMetadata{ }
\documentclass{article}
\usepackage{luamplib}
\ExplSyntaxOn
\color_model_new:nnn { pantone3005 }
{ Separation }
{
  name = PANTONE~3005~U ,
  alternative-model = cmyk ,
  alternative-values = {1, 0.56, 0, 0}
}
\color_set:nnn{spotA}{pantone3005}{1}
\color_set:nnn{spotB}{pantone3005}{0.6}
\color_model_new:nnn { pantone1215 }
{ Separation }
{
  name = PANTONE~1215~U ,
  alternative-model = cmyk ,
  alternative-values = {0, 0.15, 0.51, 0}
}
\color_set:nnn{spotC}{pantone1215}{1}
\ExplSyntaxOff
\begin{document}
\begin{mplibcode}
beginfig(1)
  fill unitsquare xscaled \mpdim\textwidth yscaled 1cm

```

```

withshadingmethod "linear"
withshadingvector (0,1)
withshadingstep (
  withshadingfraction .5
  withshadingcolors ("spotB","spotC")
)
withshadingstep (
  withshadingfraction 1
  withshadingcolors ("spotC","magenta!50")
)
;
endfig;
\end{mplibcode}
\end{document}

```

another one: user-defined DeviceN colorspace

```

\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}

687 local name, value, obj
688 if not is_xcolor(str) then
689   run_tex_code({ "\color_export:nnN{", str, "}{backend}\mplib@tempa" }, ccexplat)
690   name, value = get_macro'mplib@tempa':match'{{(.-)}{(.-)}'
691   local t = res:explode()
692   local refname = t[1]:sub(2,-1)
693   if pdfmode then

```

```

694     obj = format("%s 0 R", ltx.pdf.object_id(refname))
695   elseif t[2] == "cs" then -- raw export
696     run_tex_code({ "\\mplibmptoks\\expanded{{{\\pdf_object_ref:n{", refname, "}}}" }, cexplat)
697     obj = texgettoks"mplibmptoks"
698   else -- legacy: to be removed
699     obj = t[2]
700   end
701   else -- colorspace: mixing is allowed only in preamble
702     name, value = res:match"^(.-) cs (.-) sc"
703     name, obj = get_spc_name_obj(name)
704   end
705   return format('(1) withprescript"mplib_spotcolor=%s:%s:%s"', value,obj,name)
706 end
707 return format('(%s) withprescript"mplib_texcolor=%s"', tableconcat(colorsplit(res),""), str)
708 end
709

```

luamplib.fillandstrokecolor

```

710 do
711   local function graphicstextcolor (col, filldraw)
712     if col:find"^[%d%.:]+$" then
713       col = col:explode"."
714       for i=1,#col do
715         col[i] = format("%.3f", col[i])
716       end
717       if pdfmode then
718         local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
719         col[#col+1] = filldraw == "fill" and op or op:upper()
720         return tableconcat(col," ")
721       end
722       return format("[%s]", tableconcat(col," "))
723     end
724     col = process_color(col):match'"mpliboverridecolor=(.)"'
725     if pdfmode or col:find" cs " then
726       local t = col:explode()
727       local b = filldraw == "fill" and 1 or #t/2+1
728       local e = b == 1 and #t/2 or #t
729       return tableconcat(t," ", b, e)
730     end
731     if col:find"@pdf.obj" then
732       return col:gsub("pdf:bc%s*", "", 1)
733     else
734       return format("[%s]", tableconcat(colorsplit(col)," "))
735     end
736   end
737   function luamplib.fillandstrokecolor (fill, stroke)
738     fill = graphicstextcolor(fill, "fill")
739     stroke = graphicstextcolor(stroke, "stroke")
740     local bc = (pdfmode or fill:find" cs ") and "" or "pdf:bc "

```

```

741   return format('withprescript "mpliboverridecolor=%s%s %s"', bc, fill, stroke)
742 end
743 end
744

```

Remove trailing zeros for smaller PDF

```

745 local decimals = "%.d+"
746 local function rmzeros(str) return str:gsub("%.?0+$", "") end
747

```

common function for mplibgraphicstext and mpliboutlinetext

```

748 local function getrulemetric (box, curr, bp)
749   local running = -1073741824
750   local wd,ht,dp = curr.width, curr.height, curr.depth
751   wd = wd == running and box.width or wd
752   ht = ht == running and box.height or ht
753   dp = dp == running and box.depth or dp
754   if bp then
755     return wd/factor, ht/factor, dp/factor
756   end
757   return wd, ht, dp
758 end
759

```

luamplib's mplibgraphicstext operator

```

760 do
761   if not math.round then
762     function math.round(x) return x < 0 and -math.floor(-x + 0.5) or math.floor(x + 0.5) end
763   end
764   local emboldenfonts = { }
765   local function roundupwidth (f, fb)
766     local wd = math.round(f.size * fb / factor * 10)
767     if wd == 0 and fb ~= 0 then
768       wd = 1
769     end
770     emboldenfonts.width = wd
771     return wd
772   end
773   local function getemboldenwidth (curr, fakebold)
774     local width = emboldenfonts.width
775     if not width then
776       local f
777       local function getglyph(n)
778         while n do
779           if n.head then
780             getglyph(n.head)
781           elseif n.font and n.font > 0 then
782             f = n.font; break
783           end
784           n = node.getnext(n)

```

```

785     end
786     end
787     getglyph(curr)
788     width = roundupwidth(font.getcopy(f or font.current()), fakebold)
789     end
790     return width
791 end
792 local function getrulewhatsit (line, wd, ht, dp)
793     line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
794     line = line == 0 and "" or ("%f w"):format(line)
795     local pl
796     local fmt = "q %s %f %f %f %f re B Q"
797     if pdfmode then
798         pl = node.new("whatsit", "pdf_literal")
799         pl.mode = 0
800     else
801         fmt = "pdf:content " .. fmt
802         pl = node.new("whatsit", "special")
803     end
804     pl.data = fmt:format(line, 0, -dp, wd, ht+dp) :gsub(decimals, rmzeros)
805     local ss = node.new"glue"
806     node.setglue(ss, 0, 65536, 65536, 2, 2)
807     pl.next = ss
808     return pl
809 end

```

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

```

810 local tag_update_attrs
811 if is_defined"ver@tagpdf.sty" then
812     tag_update_attrs = function (n, curr)
813         while n do
814             n.attr = curr.attr
815             if n.head then
816                 tag_update_attrs(n.head, curr)
817             end
818             n = node.getnext(n)
819         end
820     end
821 else
822     tag_update_attrs = function() end
823 end
824 local function embolden (box, curr, fakebold)
825     local head = curr
826     while curr do
827         if curr.head then
828             curr.head = embolden(curr, curr.head, fakebold)
829         elseif curr.replace then
830             curr.replace = embolden(box, curr.replace, fakebold)
831         elseif curr.leader then

```

```

832     if curr.leader.head then
833         curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
834     elseif curr.leader.id == node.id"rule" then
835         local glue = node.effective_glue(curr, box)
836         local line = getemboldenwidth(curr, fakebold)
837         local wd,ht,dp = getrulemetric(box, curr.leader)
838         if box.id == node.id"hlist" then
839             wd = glue
840         else
841             ht, dp = 0, glue
842         end
843         local pl = getrulewhatsit(line, wd, ht, dp)
844         local pack = box.id == node.id"hlist" and node.hpack or node.vpack
845         local list = pack(pl, glue, "exactly")
846         tag_update_attrs(list,curr)
847         head = node.insert_after(head, curr, list)
848         head, curr = node.remove(head, curr)
849     end
850 elseif curr.id == node.id"rule" and curr.subtype == 0 then
851     local line = getemboldenwidth(curr, fakebold)
852     local wd,ht,dp = getrulemetric(box, curr)
853     if box.id == node.id"vlist" then
854         ht, dp = 0, ht+dp
855     end
856     local pl = getrulewhatsit(line, wd, ht, dp)
857     local list
858     if box.id == node.id"hlist" then
859         list = node.hpack(pl, wd, "exactly")
860     else
861         list = node.vpack(pl, ht+dp, "exactly")
862     end
863     tag_update_attrs(list,curr)
864     head = node.insert_after(head, curr, list)
865     head, curr = node.remove(head, curr)
866 elseif curr.id == node.id"glyph" and curr.font > 0 then
867     local f = curr.font
868     local key = format("%s:%s",f,fakebold)
869     local i = emboldenfonts[key]
870     if not i then
871         local ft = font.getfont(f) or font.getcopy(f)
872         local width = roundupwidth(ft, fakebold)
873         if ft.format == "opentype" or ft.format == "truetype" then
874             local name = ft.name:gsub("'",'):gsub('$','')
875             local t = name:gsub("^file:","):gsub("^name:","):gsub("^kpse:","):gsub("^my:",")
876             name = format('%s%sembolden=%s;',name, t:find":" and ";" or ":", fakebold)
877             _, i = fonts.constructors.readanddefine(name,ft.size)
878         elseif pdfmode then
879             local ft = table.copy(ft)
880             ft.mode, ft.width = 2, width

```

```

881         i = font.define(ft)
882     else
883         goto skip_type1
884     end
885     emboldenfonts[key] = i
886     end
887     curr.font = i
888     end
889     ::skip_type1::
890     curr = node.getnext(curr)
891     end
892     return head
893 end
894 luamplib.graphicstext = function (text, fakebold, fc, dc)
895     local fmt = process_tex_text(text):sub(1,-2)
896     local id = tonumber(fmt:match"mplibtexboxid=(%d+)")
897     emboldenfonts.width = nil
898     local box = texgetbox(id)
899     box.head = embolden(box, box.head, fakebold)
900     local colors = luamplib.fillandstrokecolor(fc, dc)
901     return format('%s %s)', fmt, colors)
902 end
903 end
904

```

luamplib's mplibglyph operator

```

905 do
906     local function mperr (str)
907         return format("hide(errmessage %q)", str)
908     end
909     local function getangle (a,b,c)
910         local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
911         if r > 180 then
912             r = r - 360
913         elseif r < -180 then
914             r = r + 360
915         end
916         return r
917     end
918     local function turning (t)
919         local r, n = 0, #t
920         for i=1,2 do
921             tableinsert(t, t[i])
922         end
923         for i=1,n do
924             r = r + getangle(t[i], t[i+1], t[i+2])
925         end
926         return r/360
927     end

```

```

928 local function glyphimage(t, fmt)
929     local q, p, r, towarn = {},{}
930     local function closepath(dots)
931         tableinsert(p, format("%scycle", dots or "--"))
932         tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
933     end
934     for i,v in ipairs(t) do
935         local cmd = v[#v]
936         local nt = t[i+1]
937         local final = not nt or nt[#nt] ~= "l" and nt[#nt] ~= "c"
938         if cmd == "m" then
939             if final then towarn = true end
940             p = {format('(%s,%s)',v[1],v[2])}
941             r = {{x=v[1],y=v[2]}}
942         else
943             if cmd == "l" then
944                 local pt = t[i-1]
945                 if (final or pt and pt[#pt] == "m") and r[1].x == v[1] and r[1].y == v[2] then
946                     else
947                         tableinsert(p, format('--(%s,%s)',v[1],v[2]))
948                         tableinsert(r, {x=v[1],y=v[2]})
949                     end
950                 if final then closepath() end
951             elseif cmd == "c" then
952                 tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
953                 if final and r[1].x == v[5] and r[1].y == v[6] then
954                     closepath ".."
955                 else
956                     tableinsert(p, format('..(%s,%s)',v[5],v[6]))
957                     tableinsert(r, {x=v[5],y=v[6]})
958                     if final then closepath() end
959                 end
960             elseif cmd == "path" or cmd == "move" then
961                 else
962                     return mperr"unknown operator"
963                 end
964             end
965         end
966         r = { }
967         if fmt == "opentype" then
968             for _,v in ipairs(q[1]) do
969                 tableinsert(r, format('addto currentpicture contour %s;',v))
970             end
971             for _,v in ipairs(q[2]) do
972                 tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
973             end
974         else
975             for _,v in ipairs(q[2]) do
976                 tableinsert(r, format('addto currentpicture contour %s;',v))

```

```

977     end
978     for _,v in ipairs(q[1]) do
979         tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
980     end
981 end
982 return format('image(%s)', tableconcat(r)), towarn
983 end
984 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
985 function luamplib.glyph (f, c)
986     local filename, subfont, instance, kind, shapedata
987     local fid = tonumber(f) or font.id(f)
988     if fid > 0 then
989         local fontdata = font.getfont(fid) or font.getcopy(fid)
990         filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
991         instance = fontdata.specification and fontdata.specification.instance
992                 or fontdata.shared and fontdata.shared.features.axis
993         filename = filename and filename:gsub("^harfloaded:", "")
994     else
995         local name
996         f = f:match"^%s*(.)%s*$"
997         name, subfont, instance = f:match"(.+)((%d+)%)[(-)%]$"
998         if not name then
999             name, instance = f:match"(.+)%[(-)%]$" -- SourceHanSansK-VF.otf[Heavy]
1000         end
1001         if not name then
1002             name, subfont = f:match"(.+)((%d+)%)" -- Times.ttc(2)
1003         end
1004         name = name or f
1005         subfont = (subfont or 0)+1
1006         instance = instance and instance:lower()
1007         for _,ftype in ipairs{"opentype", "truetype"} do
1008             filename = kpse.find_file(name, ftype.." fonts")
1009             if filename then
1010                 kind = ftype; break
1011             end
1012         end
1013     end
1014     if kind ~= "opentype" and kind ~= "truetype" then
1015         f = fid and fid > 0 and tex.fontname(fid) or f
1016         if kpse.find_file(f, "tfm") then
1017             return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
1018         else
1019             filename = kpse.find_file(f, "type1 fonts")
1020             if filename then
1021                 kind = "type1" -- there's bug in processing cmr family
1022             else
1023                 return mperr"font not found"
1024             end
1025         end

```

```

1026     end
1027     local time = lfsattributes(filename,"modification")

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

1028     local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
1029     local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
1030     local newname = format("%s/%s.lua", cachedir or outputdir(), h)
1031     local newtime = lfsattributes(newname,"modification") or 0
1032     if time == newtime then
1033         shapedata = require(newname)
1034     end
1035     if not shapedata then
1036         if fonts then
1037             local handler = kind == "type1" and fonts.handlers.afm or fonts.handlers.otf
1038             shapedata = handler.readers.loadshapes(filename,subfont,instance)
1039         end
1040         if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
1041         table.tofile(newname, shapedata, "return")
1042         lfstouch(newname, time, time)
1043     end
1044     local gid = tonumber(c)
1045     if not gid then
1046         local uni = utf8.codepoint(c)
1047         for i,v in pairs(shapedata.glyphs) do
1048             if c == v.name or uni == v.unicode then
1049                 gid = i; break
1050             end
1051         end
1052     end
1053     if not gid then return mperr"cannot get GID (glyph id)" end
1054     local fac = 1000 / (shapedata.units or 1000)
1055     local t = shapedata.glyphs[gid]; t = t and t.segments
1056     if not t then return "image()" end
1057     for i,v in ipairs(t) do
1058         if type(v) == "table" then
1059             for ii,vv in ipairs(v) do
1060                 if type(vv) == "number" then
1061                     t[i][ii] = format("%.0f", vv * fac)
1062                 end
1063             end
1064         end
1065     end
1066     local result, towarn = glyphimage(t, shapedata.format or kind)
1067     if towarn then
1068         warn("mplibglyph %s not working properly. Use glyph instead", f)
1069     end
1070     return result

```

```

1071 end
1072 end
1073

```

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

```

1074 do
1075 local rulefmt = "mpliboutlinepic[%i]:=image(addto currentpicture contour \z
1076 unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
1077 local outline_horz, outline_vert
1078 function outline_vert (res, box, curr, xshift, yshift)
1079 local b2u = box.dir == "LTL"
1080 local dy = (b2u and -box.depth or box.height)/factor
1081 local ody = dy
1082 while curr do
1083 if curr.id == node.id"rule" then
1084 local wd, ht, dp = getrulemetric(box, curr, true)
1085 local hd = ht + dp
1086 if hd ~= 0 then
1087 dy = dy + (b2u and dp or -ht)
1088 if wd ~= 0 and curr.subtype == 0 then
1089 res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
1090 end
1091 dy = dy + (b2u and ht or -dp)
1092 end
1093 elseif curr.id == node.id"glue" then
1094 local vwidth = node.effective_glue(curr,box)/factor
1095 if curr.leader then
1096 local curr, kind = curr.leader, curr.subtype
1097 if curr.id == node.id"rule" then
1098 local wd = getrulemetric(box, curr, true)
1099 if wd ~= 0 then
1100 local hd = vwidth
1101 local dy = dy + (b2u and 0 or -hd)
1102 if hd ~= 0 and curr.subtype == 0 then
1103 res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
1104 end
1105 end
1106 elseif curr.head then
1107 local hd = (curr.height + curr.depth)/factor
1108 if hd <= vwidth then
1109 local dy, n, iy = dy, 0, 0
1110 if kind == 100 or kind == 103 then -- todo: gleaders
1111 local ady = abs(ody - dy)
1112 local ndy = math.ceil(ady / hd) * hd
1113 local diff = ndy - ady
1114 n = math.floor((vwidth-diff) / hd)
1115 dy = dy + (b2u and diff or -diff)
1116 else
1117 n = math.floor(vwidth / hd)

```

```

1118         if kind == 101 then
1119             local side = vwidth % hd / 2
1120             dy = dy + (b2u and side or -side)
1121         elseif kind == 102 then
1122             iy = vwidth % hd / (n+1)
1123             dy = dy + (b2u and iy or -iy)
1124         end
1125     end
1126     dy = dy + (b2u and curr.depth or -curr.height)/factor
1127     hd = b2u and hd or -hd
1128     iy = b2u and iy or -iy
1129     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1130     for i=1,n do
1131         res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1132         dy = dy + hd + iy
1133     end
1134 end
1135 end
1136 end
1137 dy = dy + (b2u and vwidth or -vwidth)
1138 elseif curr.id == node.id"kern" then
1139     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1140 elseif curr.id == node.id"vlist" then
1141     dy = dy + (b2u and curr.depth or -curr.height)/factor
1142     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1143     dy = dy + (b2u and curr.height or -curr.depth)/factor
1144 elseif curr.id == node.id"hlist" then
1145     dy = dy + (b2u and curr.depth or -curr.height)/factor
1146     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1147     dy = dy + (b2u and curr.height or -curr.depth)/factor
1148 end
1149 curr = node.getnext(curr)
1150 end
1151 return res
1152 end
1153 function outline_horz (res, box, curr, xshift, yshift, discwd)
1154     local r2l = box.dir == "TRT"
1155     local dx = r2l and (discwd or box.width/factor) or 0
1156     local dirs = { { dir = r2l, dx = dx } }
1157     while curr do
1158         if curr.id == node.id"dir" then
1159             local sign, dir = curr.dir:match"(.)(...)"
1160             local level, newdir = curr.level, r2l
1161             if sign == "+" then
1162                 newdir = dir == "TRT"
1163             if r2l ~= newdir then
1164                 local n = node.getnext(curr)
1165                 while n do
1166                     if n.id == node.id"dir" and n.level+1 == level then break end

```

```

1167         n = node.getnext(n)
1168     end
1169     n = n or node.tail(curr)
1170     dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1171     end
1172     dirs[level] = { dir = r2l, dx = dx }
1173 else
1174     local level = level + 1
1175     newdir = dirs[level].dir
1176     if r2l ~= newdir then
1177         dx = dirs[level].dx
1178     end
1179 end
1180 r2l = newdir
1181 elseif curr.char and curr.font and curr.font > 0 then
1182     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1183     local gid = ft.characters[curr.char].index or curr.char
1184     local scale = ft.size / factor / 1000
1185     local slant = (ft.slant or 0)/1000
1186     local extend = (ft.extend or 1000)/1000
1187     local squeeze = (ft.squeeze or 1000)/1000
1188     local expand = 1 + (curr.expansion_factor or 0)/1000000
1189     local xscale, yscale = scale * extend * expand, scale * squeeze
1190     dx = dx - (r2l and curr.width/factor*expand or 0)
1191     local xoff, yoff = (curr.xoffset or 0)/factor, (curr.yoffset or 0)/factor
1192     local xpos, ypos = dx + xshift + xoff, yshift + yoff
1193     local vertical = ""
1194     if ft.shared and (ft.shared.features.vert or ft.shared.features.vrt2) then
1195         if ft.shared.features.vertical then -- luatexko
1196             vertical = "rotated 90"
1197             local data = ft.characters[curr.char] or { }
1198             if ft.hb then
1199                 local hoff, voff = (data.luatexko_hoff or 0)/factor, (data.luatexko_voff or 0)/factor
1200                 local charraise = (ft.luatexko_charraise or 0)/factor
1201                 xpos, ypos = xpos - voff + hoff - charraise, ypos + hoff + voff + charraise
1202             else
1203                 local cmds = data.commands or { {0,0}, {0,0} }
1204                 local voff, hoff = -cmds[1][2]/factor, cmds[2][2]/factor
1205                 xpos, ypos = xpos + hoff, ypos + voff
1206             end
1207         elseif curr ~= box.head then -- luatexja
1208             vertical = "rotated 90"
1209             local en = ft.parameters.quad/factor/2
1210             xpos, ypos = xpos - xoff - yoff + en, ypos - yoff + xoff - en
1211         end
1212     end
1213     local image
1214     if ft.format == "opentype" or ft.format == "truetype" then
1215         image = luampplib.glyph(curr.font, gid)

```

```

1216     else
1217         local name, scale = ft.name, 1
1218         local vf = font.read_vf(name, ft.size)
1219         if vf and vf.characters[gid] then
1220             local cmds = vf.characters[gid].commands or {}
1221             for _,v in ipairs(cmds) do
1222                 if v[1] == "char" then
1223                     gid = v[2]
1224                 elseif v[1] == "font" and vf.fonts[v[2]] then
1225                     name = vf.fonts[v[2]].name
1226                     scale = vf.fonts[v[2]].size / ft.size
1227                 end
1228             end
1229         end
1230         image = format("glyph %s of %q scaled %f", gid, name, scale)
1231     end
1232     res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1233                         #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1234     dx = dx + (r2l and 0 or curr.width/factor*expand)
1235 elseif curr.replace then
1236     local width = node.dimensions(curr.replace)/factor
1237     dx = dx - (r2l and width or 0)
1238     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1239     dx = dx + (r2l and 0 or width)
1240 elseif curr.id == node.id"rule" then
1241     local wd, ht, dp = getrulemetric(box, curr, true)
1242     if wd ~= 0 then
1243         local hd = ht + dp
1244         dx = dx - (r2l and wd or 0)
1245         if hd ~= 0 and curr.subtype == 0 then
1246             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1247         end
1248         dx = dx + (r2l and 0 or wd)
1249     end
1250 elseif curr.id == node.id"glue" then
1251     local width = node.effective_glue(curr, box)/factor
1252     dx = dx - (r2l and width or 0)
1253     if curr.leader then
1254         local curr, kind = curr.leader, curr.subtype
1255         if curr.id == node.id"rule" then
1256             local wd, ht, dp = getrulemetric(box, curr, true)
1257             local hd = ht + dp
1258             if hd ~= 0 then
1259                 wd = width
1260                 if wd ~= 0 and curr.subtype == 0 then
1261                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1262                 end
1263             end
1264         elseif curr.head then

```

```

1265     local wd = curr.width/factor
1266     if wd <= width then
1267         local dx = r2l and dx+width or dx
1268         local n, ix = 0, 0
1269         if kind == 100 or kind == 103 then -- todo: gleaders
1270             local adx = abs(dx-dirs[1].dx)
1271             local ndx = math.ceil(adx / wd) * wd
1272             local diff = ndx - adx
1273             n = math.floor((width-diff) / wd)
1274             dx = dx + (r2l and -diff-wd or diff)
1275         else
1276             n = math.floor(width / wd)
1277             if kind == 101 then
1278                 local side = width % wd / 2
1279                 dx = dx + (r2l and -side-wd or side)
1280             elseif kind == 102 then
1281                 ix = width % wd / (n+1)
1282                 dx = dx + (r2l and -ix-wd or ix)
1283             end
1284         end
1285         wd = r2l and -wd or wd
1286         ix = r2l and -ix or ix
1287         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1288         for i=1,n do
1289             res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1290             dx = dx + wd + ix
1291         end
1292     end
1293 end
1294 end
1295 dx = dx + (r2l and 0 or width)
1296 elseif curr.id == node.id"kern" then
1297     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1298 elseif curr.id == node.id"math" then
1299     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1300 elseif curr.id == node.id"vlist" then
1301     dx = dx - (r2l and curr.width/factor or 0)
1302     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1303     dx = dx + (r2l and 0 or curr.width/factor)
1304 elseif curr.id == node.id"hlist" then
1305     dx = dx - (r2l and curr.width/factor or 0)
1306     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1307     dx = dx + (r2l and 0 or curr.width/factor)
1308 end
1309 curr = node.getnext(curr)
1310 end
1311 return res
1312 end
1313 function luamplib.outlinetext (text)

```

```

1314 local fmt = process_tex_text(text)
1315 local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1316 local box = texgetbox(id)
1317 local res = outline_horz({ }, box, box.head, 0, 0)
1318 if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1319 return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1320 end
1321 end
1322

```

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

```

1323 function luamplib.getunicodegraphemes (s)
1324 local t = { }
1325 local graphemes = require'lua-uni-graphemes'
1326 for _, _, c in graphemes.graphemes(s) do
1327 table.insert(t, c)
1328 end
1329 return t
1330 end
1331 function luamplib.unicodesubstring (s,b,e,grph)
1332 local tt, t, step = { }
1333 if grph then
1334 t = luamplib.getunicodegraphemes(s)
1335 else
1336 t = { }
1337 for _, c in utf8.codes(s) do
1338 table.insert(t, utf8.char(c))
1339 end
1340 end
1341 if b <= e then
1342 b, step = b+1, 1
1343 else
1344 e, step = e+1, -1
1345 end
1346 for i = b, e, step do
1347 table.insert(tt, t[i])
1348 end
1349 s = table.concat(tt):gsub("'", "'&ditto'")
1350 return string.format("%s", s)
1351 end
1352

```

METAPOST preambles

```

1353 luamplib.preambles = {
1354 preamble = [[
1355 boolean mplib ; mplib := true ;
1356 let dump = endinput ;
1357 let normalfontsize = fontsize;
1358 input %s ;
1359 ]],

```

```

1360 mplibcode = [[
1361 texscriptmode := 2;
1362 def rawtexttext primary t = runscript("luamplibtext{"&t&}") enddef;
1363 def mplibcolor primary t = runscript("luamplibcolor{"&t&}") enddef;
1364 def mplibdimen primary t = runscript("luamplibdimen{"&t&}") enddef;
1365 def VerbatimTeX primary t = runscript("luamplibverbtex{"&t&}") enddef;
1366 if known context_mlib:
1367   defaultfont := "cmtt10";
1368   let infont = normalinfont;
1369   let fontsize = normalfontsize;
1370   vardef thelabel@#(expr p,z) =
1371     if string p :
1372       thelabel@#(p infont defaultfont scaled defaultscale,z)
1373     else :
1374       p shifted (z + labeloffset*mfun_laboff@# -
1375         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1376         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1377     fi
1378   enddef;
1379 else:
1380   vardef texttext@# primary t = rawtexttext (t) enddef;
1381   def message expr t =
1382     if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1383   enddef;
1384   def withtransparency (expr a, t) =
1385     withprescript "tr_alternative=" & if numeric a: decimal fi a
1386     withprescript "tr_transparency=" & decimal t
1387   enddef;
1388   vardef ddecimal primary p =
1389     decimal xpart p & " " & decimal ypart p
1390   enddef;
1391   vardef boundingbox primary p =
1392     if (path p) or (picture p) :
1393       llcorner p -- lrcorner p -- urcorner p -- ulcorner p
1394     else :
1395       origin
1396     fi -- cycle
1397   enddef;
1398 fi
1399 def resolvedcolor(expr s) =
1400   runscript("return luamplib.shadecolor('"&s &"')")
1401 enddef;
1402 def colordecimals primary c =
1403   if cmykcolor c:
1404     decimal cyanpart c & ":" & decimal magentapart c & ":" &
1405     decimal yellowpart c & ":" & decimal blackpart c
1406   elseif rgbcolor c:
1407     decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1408   elseif string c:

```

```

1409   if known graphicstextpic: c else: colordecimals resolvedcolor(c) fi
1410   else:
1411     decimal c
1412   fi
1413 enddef;
1414 def externalfigure primary filename =
1415   draw rawtexttext("\includegraphics{"& filename &}")
1416 enddef;
1417 def TEX = texttext enddef;
1418 def mplibtexcolor primary c =
1419   runscript("return luamplib.gettexcolor('"& c & "')")
1420 enddef;
1421 def mplibrgbtexcolor primary c =
1422   runscript("return luamplib.gettexcolor('"& c & "', 'rgb')")
1423 enddef;
1424 def mplibgraphicstext primary t =
1425   begingroup;
1426   mplibgraphicstext_ (t)
1427 enddef;
1428 def mplibgraphicstext_ (expr t) text rest =
1429   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor, strokecolor,
1430   fb, fc, dc, graphicstextpic, alsoordoublepath;
1431   picture graphicstextpic; graphicstextpic := nullpicture;
1432   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1433   let scale = scaled;
1434   def fakebold primary c = hide(fb:=c;) enddef;
1435   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1436   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1437   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1438   def alsoordoublepath expr p = if picture p: also else: doublepath fi p enddef;
1439   addto graphicstextpic alsoordoublepath (origin--cycle) rest; graphicstextpic:=nullpicture;
1440   def fakebold primary c = enddef;
1441   let fillcolor = fakebold; let drawcolor = fakebold;
1442   let withfillcolor = fillcolor; let withdrawcolor = drawcolor; let strokecolor = drawcolor;
1443   image(draw runscript("return luamplib.graphicstext([===["&t&"]===], "
1444     & decimal fb & ", '"& fc & "', '"& dc & "')") rest;)
1445   endgroup;
1446 enddef;
1447 def mplibglyph expr c of f =
1448   runscript (
1449     "return luamplib.glyph('"
1450     & if numeric f: decimal fi f
1451     & "' ,'"
1452     & if numeric c: decimal fi c
1453     & "')"
1454   )
1455 enddef;
1456 numeric luamplib_tmp_num_; luamplib_tmp_num_ = 0;
1457 def mplibdrawglyph expr g =

```

```

1458 luamplib_tmp_num_ := 0;
1459 for item within g:
1460   fill pathpart item
1461   if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1462 endfor
1463 enddef;
1464 let mplibfillglyph = mplibdrawglyph;
1465 def mplibstrokeglyph expr g =
1466   luamplib_tmp_num_ := 0;
1467   for item within g:
1468     draw pathpart item
1469     if incr luamplib_tmp_num_ < length g: withpostscript "collect"; fi
1470   endfor
1471 enddef;
1472 def mplibfillandstrokeglyph expr g =
1473   luamplib_tmp_num_ := 0;
1474   for item within g:
1475     draw pathpart item withpostscript
1476     if incr luamplib_tmp_num_ < length g: "collect"; else: "both" fi
1477   endfor
1478 enddef;
1479 def withmplibcolors (expr f, s) =
1480   runscript("return luamplib.fillandstrokecolor('" &
1481     if not string f: colordecimals fi f & "'','" &
1482     if not string s: colordecimals fi s & "'')")
1483 enddef;
1484 def withmplibopacities (expr a, f, s) =
1485   withprescript "tr_alternative=" & if numeric a: decimal fi a
1486   withprescript "tr_transparency=" & decimal f & ":" & decimal s
1487 enddef;
1488 def mplib_do_outline_text_set_b (text f) (text d) text r =
1489   def mplib_do_outline_options_f = f enddef;
1490   def mplib_do_outline_options_d = d enddef;
1491   def mplib_do_outline_options_r = r enddef;
1492 enddef;
1493 def mplib_do_outline_text_set_f (text f) text r =
1494   def mplib_do_outline_options_f = f enddef;
1495   def mplib_do_outline_options_r = r enddef;
1496 enddef;
1497 def mplib_do_outline_text_set_u (text f) text r =
1498   def mplib_do_outline_options_f = f enddef;
1499 enddef;
1500 def mplib_do_outline_text_set_d (text d) text r =
1501   def mplib_do_outline_options_d = d enddef;
1502   def mplib_do_outline_options_r = r enddef;
1503 enddef;
1504 def mplib_do_outline_text_set_r (text d) (text f) text r =
1505   def mplib_do_outline_options_d = d enddef;
1506   def mplib_do_outline_options_f = f enddef;

```

```

1507 def mplib_do_outline_options_r = r enddef;
1508 enddef;
1509 def mplib_do_outline_text_set_n text r =
1510   def mplib_do_outline_options_r = r enddef;
1511 enddef;
1512 def mplib_do_outline_text_set_p = enddef;
1513 def mplib_fill_outline_text =
1514   for n=1 upto mpliboutlinenum:
1515     i:=0;
1516     for item within mpliboutlinepic[n]:
1517       i:=i+1;
1518       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1519       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1520     endfor
1521   endfor
1522 enddef;
1523 def mplib_draw_outline_text =
1524   for n=1 upto mpliboutlinenum:
1525     for item within mpliboutlinepic[n]:
1526       draw pathpart item mplib_do_outline_options_d;
1527     endfor
1528   endfor
1529 enddef;
1530 def mplib_filldraw_outline_text =
1531   for n=1 upto mpliboutlinenum:
1532     i:=0;
1533     for item within mpliboutlinepic[n]:
1534       i:=i+1;
1535       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1536         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1537       else:
1538         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1539       fi
1540     endfor
1541   endfor
1542 enddef;
1543 vardef mpliboutlinetext@# (expr t) text rest =
1544   save kind; string kind; kind := str @#;
1545   save i; numeric i;
1546   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1547   def mplib_do_outline_options_d = enddef;
1548   def mplib_do_outline_options_f = enddef;
1549   def mplib_do_outline_options_r = enddef;
1550   runscript("return luamplib.outlinetext[===["&t&"]===");
1551   image ( addto currentpicture also image (
1552     if kind = "f":
1553       mplib_do_outline_text_set_f rest;
1554       mplib_fill_outline_text;
1555     elseif kind = "d":

```

```

1556     mplib_do_outline_text_set_d rest;
1557     mplib_draw_outline_text;
1558     elseif kind = "b":
1559         mplib_do_outline_text_set_b rest;
1560         mplib_fill_outline_text;
1561         mplib_draw_outline_text;
1562     elseif kind = "u":
1563         mplib_do_outline_text_set_u rest;
1564         mplib_filldraw_outline_text;
1565     elseif kind = "r":
1566         mplib_do_outline_text_set_r rest;
1567         mplib_draw_outline_text;
1568         mplib_fill_outline_text;
1569     elseif kind = "p":
1570         mplib_do_outline_text_set_p;
1571         mplib_draw_outline_text;
1572     else:
1573         mplib_do_outline_text_set_n rest;
1574         mplib_fill_outline_text;
1575     fi;
1576 ) mplib_do_outline_options_r; )
1577 enddef ;
1578 def withmppattern primary p =
1579     withprescript "mplibpattern=" & if numeric p: decimal fi p
1580 enddef;
1581 primarydef t withpattern p =
1582     image(
1583         if cycle t:
1584             fill
1585         else:
1586             draw
1587         fi
1588         t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1589 enddef;
1590 vardef mplibtransformmatrix (text e) =
1591     save t; transform t;
1592     t = identity e;
1593     runscript("luamplib.transformmatrix = {"
1594         & decimal xpart t & ","
1595         & decimal ypart t & ","
1596         & decimal xpart t & ","
1597         & decimal ypart t & ","
1598         & decimal xpart t & ","
1599         & decimal ypart t & ","
1600         & "}");
1601 enddef;
1602 primarydef p withmaskinggroup s =
1603     if picture p:
1604         image(

```

```

1605     draw p;
1606     draw center p withprescript "mplibfadestate=stop";
1607 )
1608 else:
1609   p withprescript "mplibfadestate=stop"
1610 fi
1611 withprescript "mplibfadetype=masking"
1612 withprescript "mplibmaskname=" & s
1613 enddef;
1614 def withmaskingbgcolor expr c =
1615   withprescript "mplibmaskingbgcolor=" & colordecimals c
1616 enddef;
1617 primarydef p withfademethod s =
1618   if picture p:
1619     image(
1620       draw p;
1621       draw center p withprescript "mplibfadestate=stop";
1622     )
1623   else:
1624     p withprescript "mplibfadestate=stop"
1625   fi
1626   withprescript "mplibfadetype=" & s
1627   hide(mplib_shade_step := 1;)
1628   withprescript "sh_color_a=1"
1629   withprescript "sh_color_b=0"
1630   withprescript "mplibfadebbox=" &
1631     decimal (xpart llcorner p -1/4) & ":" &
1632     decimal (ypart llcorner p -1/4) & ":" &
1633     decimal (xpart urcorner p +1/4) & ":" &
1634     decimal (ypart urcorner p +1/4)
1635 enddef;
1636 def withfadevector (expr a,b) =
1637   withprescript "mplibfadevector=" &
1638     decimal xpart a & ":" &
1639     decimal ypart a & ":" &
1640     decimal xpart b & ":" &
1641     decimal ypart b
1642 enddef;
1643 let withfadecenter = withfadevector;
1644 def withfaderadius (expr a,b) =
1645   withprescript "mplibfaderadius=" &
1646     decimal a & ":" &
1647     decimal b
1648 enddef;
1649 def withfadebbox (expr a,b) =
1650   withprescript "mplibfadebbox=" &
1651     decimal xpart a & ":" &
1652     decimal ypart a & ":" &
1653     decimal xpart b & ":" &

```

```

1654     decimal ypart b
1655 enddef;
1656 primarydef p asgroup s =
1657   image(
1658     draw center p
1659     withprescript "mplibgroupbbox=" &
1660     decimal (xpart llcorner p -1/4) & ":" &
1661     decimal (ypart llcorner p -1/4) & ":" &
1662     decimal (xpart urcorner p +1/4) & ":" &
1663     decimal (ypart urcorner p +1/4)
1664     withprescript "gr_state=start"
1665     withprescript "gr_type=" & s;
1666     draw p withprescript "sh_in_xobj=yes";
1667     draw center p withprescript "gr_state=stop";
1668   )
1669 enddef;
1670 def withgroupbbox (expr a,b) =
1671   withprescript "mplibgroupbbox=" &
1672   decimal xpart a & ":" &
1673   decimal ypart a & ":" &
1674   decimal xpart b & ":" &
1675   decimal ypart b
1676 enddef;
1677 def withgroupname expr s =
1678   withprescript "mplibgroupname=" & s
1679 enddef;
1680 def usemplibgroup primary s =
1681   draw maketext("\luamplibtagasgroupput{"& s &"}{\csname luamplib.group."& s & "\endcsname}")
1682   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1683 enddef;
1684 path    mplib_shade_path ;
1685 numeric mplib_shade_step ; mplib_shade_step := 0 ;
1686 numeric mplib_shade_fx, mplib_shade_fy ;
1687 numeric mplib_shade_lx, mplib_shade_ly ;
1688 numeric mplib_shade_nx, mplib_shade_ny ;
1689 numeric mplib_shade_dx, mplib_shade_dy ;
1690 numeric mplib_shade_tx, mplib_shade_ty ;
1691 primarydef p withshadingmethod m =
1692   p
1693   if picture p :
1694     withprescript "sh_operand_type=picture"
1695     if textual p or (length p > 1):
1696       withprescript "sh_transform=no"
1697       mplib_with_shade_method (boundingbox p, m)
1698     else:
1699       withprescript "sh_transform=yes"
1700       mplib_with_shade_method (pathpart p, m)
1701     fi
1702   else :

```

```

1703   withprescript "sh_transform=yes"
1704   mplib_with_shade_method (p, m)
1705   fi
1706 enddef;
1707 def mplib_with_shade_method (expr p, m) =
1708   hide(mplib_with_shade_method_analyze(p))
1709   withprescript "sh_domain=0 1"
1710   withprescript "sh_color=into"
1711   withprescript "sh_color_a=" & colordecimals white
1712   withprescript "sh_color_b=" & colordecimals black
1713   withprescript "sh_first=" & ddecimal point 0 of p
1714   withprescript "sh_set_x=" & ddecimal (mplib_shade_nx,mplib_shade_lx)
1715   withprescript "sh_set_y=" & ddecimal (mplib_shade_ny,mplib_shade_ly)
1716   if m = "linear" :
1717     withprescript "sh_type=linear"
1718     withprescript "sh_factor=1"
1719     withprescript "sh_center_a=" & ddecimal llcorner p
1720     withprescript "sh_center_b=" & ddecimal urcorner p
1721   elseif m = "coons":
1722     withprescript "sh_type=coons"
1723     withprescript "sh_transform=no"
1724   elseif m = "triangle":
1725     withprescript "sh_type=triangle"
1726     withprescript "sh_transform=no"
1727   elseif m = "lattice":
1728     withprescript "sh_type=lattice"
1729     withprescript "sh_transform=no"
1730   elseif m = "tensor":
1731     withprescript "sh_type=tensor"
1732     withprescript "sh_transform=no"
1733     withprescript "sh_tensor_path=" &
1734       ddecimal 1/4[point 0 of p, point 2 of p] & " " &
1735       ddecimal 1/4[point 1 of p, point 3 of p] & " " &
1736       ddecimal 1/4[point 2 of p, point 0 of p] & " " &
1737       ddecimal 1/4[point 3 of p, point 1 of p]
1738   else :
1739     withprescript "sh_type=circular"
1740     withprescript "sh_factor=1.2"
1741     withprescript "sh_center_a=" & ddecimal center p
1742     withprescript "sh_center_b=" & ddecimal center p
1743     withprescript "sh_radius_a=" & decimal 0
1744     withprescript "sh_radius_b=" & decimal mplib_max_radius(p)
1745   fi
1746 enddef;
1747 def withlatticeverticesperrow expr n =
1748   withprescript "sh_lattice_perrow=" & decimal n
1749 enddef;
1750 def withlatticeverticesdata (text t) =
1751   hide(luamplib_tmp_num_ := 0;)

```

```

1752 withprescript "sh_lattice_data=" &
1753 for item = t:
1754   if odd(incr luamplib_tmp_num_):
1755     if pair item: ddecimal fi item & " " &
1756     fi
1757   endfor ""
1758   hide(luamplib_tmp_num_ := 0; mplib_shade_step := 0;)
1759   for item = t:
1760     if not odd(incr luamplib_tmp_num_):
1761       withshadingstep( withshadingcolors(item,item) )
1762     fi
1763   endfor
1764   enddef;
1765   def withtrianglepatchinit (expr p, a, b, c) =
1766     if string p:
1767       withtrianglepatchnext(0, p , a)
1768       withtrianglepatchnext(0, "", b)
1769       withtrianglepatchnext(0, "", c)
1770     else:
1771       withtrianglepatchnext(0, point 0 of p, a)
1772       withtrianglepatchnext(0, point 1 of p, b)
1773       withtrianglepatchnext(0, point 2 of p, c)
1774     fi
1775   enddef;
1776   def withtrianglepatchnext (expr f, p, a) =
1777     withshadingstep(
1778       withprescript "sh_triangle_edge_" & decimal mplib_shade_step & "=" & decimal f
1779       withprescript "sh_triangle_vertex_" & decimal mplib_shade_step & "=" &
1780       if string p: p else: ddecimal p fi
1781       withshadingcolors (a, a)
1782     )
1783   enddef;
1784   def withcoonspatchinit (expr p, a, b, c, d) =
1785     withshadingstep(
1786       withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1787       withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1788       if string p: p
1789       else:
1790         ddecimal point      0 of p & " " &
1791         ddecimal postcontrol 0 of p & " " &
1792         ddecimal precontrol  1 of p & " " &
1793         ddecimal point      1 of p & " " &
1794         ddecimal postcontrol 1 of p & " " &
1795         ddecimal precontrol  2 of p & " " &
1796         ddecimal point      2 of p & " " &
1797         ddecimal postcontrol 2 of p & " " &
1798         ddecimal precontrol  3 of p & " " &
1799         ddecimal point      3 of p & " " &
1800         ddecimal postcontrol 3 of p & " " &

```

```

1801     ddecimal precontrol 0 of p
1802     fi
1803     withshadingcolors (a, b)
1804 )
1805 withshadingstep ( withshadingcolors (c, d) )
1806 enddef;
1807 def withtensorpatchinit (expr p, q, a, b, c, d) =
1808   withshadingstep(
1809     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=0"
1810     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1811     if string p: p & " " & q
1812     else:
1813       ddecimal point      0 of p & " " &
1814       ddecimal postcontrol 0 of p & " " &
1815       ddecimal precontrol  1 of p & " " &
1816       ddecimal point      1 of p & " " &
1817       ddecimal postcontrol 1 of p & " " &
1818       ddecimal precontrol  2 of p & " " &
1819       ddecimal point      2 of p & " " &
1820       ddecimal postcontrol 2 of p & " " &
1821       ddecimal precontrol  3 of p & " " &
1822       ddecimal point      3 of p & " " &
1823       ddecimal postcontrol 3 of p & " " &
1824       ddecimal precontrol  0 of p & " " &
1825       ddecimal point      0 of q & " " &
1826       ddecimal point      1 of q & " " &
1827       ddecimal point      2 of q & " " &
1828       ddecimal point      3 of q
1829     fi
1830     withshadingcolors (a, b)
1831 )
1832 withshadingstep ( withshadingcolors (c, d) )
1833 enddef;
1834 def withcoonspatchnext (expr f, p, a, b) =
1835   withshadingstep(
1836     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1837     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1838     if string p: p
1839     else:
1840       ddecimal postcontrol 1 of p & " " &
1841       ddecimal precontrol  2 of p & " " &
1842       ddecimal point      2 of p & " " &
1843       ddecimal postcontrol 2 of p & " " &
1844       ddecimal precontrol  3 of p & " " &
1845       ddecimal point      3 of p & " " &
1846       ddecimal postcontrol 3 of p & " " &
1847       ddecimal precontrol  0 of p
1848     fi
1849     withshadingcolors (a, b)

```

```

1850 )
1851 endif;
1852 def withtensorpatchnext (expr f, p, q, a, b) =
1853   withshadingstep(
1854     withprescript "sh_coons_edge_" & decimal mplib_shade_step & "=" & decimal f
1855     withprescript "sh_coons_path_" & decimal mplib_shade_step & "=" &
1856       if string p: p & " " & q
1857         else:
1858           ddecimal postcontrol 1 of p & " " &
1859           ddecimal precontrol 2 of p & " " &
1860           ddecimal point 2 of p & " " &
1861           ddecimal postcontrol 2 of p & " " &
1862           ddecimal precontrol 3 of p & " " &
1863           ddecimal point 3 of p & " " &
1864           ddecimal postcontrol 3 of p & " " &
1865           ddecimal precontrol 0 of p & " " &
1866           ddecimal point 0 of q & " " &
1867           ddecimal point 1 of q & " " &
1868           ddecimal point 2 of q & " " &
1869           ddecimal point 3 of q
1870         fi
1871     withshadingcolors (a, b)
1872   )
1873 endif;
1874 def mplib_with_shade_method_analyze(expr p) =
1875   mplib_shade_path := p ;
1876   mplib_shade_step := 1 ;
1877   mplib_shade_fx := xpart point 0 of p ;
1878   mplib_shade_fy := ypart point 0 of p ;
1879   mplib_shade_lx := mplib_shade_fx ;
1880   mplib_shade_ly := mplib_shade_fy ;
1881   mplib_shade_nx := 0 ;
1882   mplib_shade_ny := 0 ;
1883   mplib_shade_dx := abs(mplib_shade_fx - mplib_shade_lx) ;
1884   mplib_shade_dy := abs(mplib_shade_fy - mplib_shade_ly) ;
1885   for i=1 upto length(p) :
1886     mplib_shade_tx := abs(mplib_shade_fx - xpart point i of p) ;
1887     mplib_shade_ty := abs(mplib_shade_fy - ypart point i of p) ;
1888     if mplib_shade_tx > mplib_shade_dx :
1889       mplib_shade_nx := i + 1 ;
1890       mplib_shade_lx := xpart point i of p ;
1891       mplib_shade_dx := mplib_shade_tx ;
1892     fi ;
1893     if mplib_shade_ty > mplib_shade_dy :
1894       mplib_shade_ny := i + 1 ;
1895       mplib_shade_ly := ypart point i of p ;
1896       mplib_shade_dy := mplib_shade_ty ;
1897     fi ;
1898   endfor ;

```

```

1899 enddef;
1900 vardef mplib_max_radius(expr p) =
1901   max (
1902     (xpart center p - xpart llcorner p) ++ (ypart center p - ypart llcorner p),
1903     (xpart center p - xpart ulcorner p) ++ (ypart ulcorner p - ypart center p),
1904     (xpart lrcorner p - xpart center p) ++ (ypart center p - ypart lrcorner p),
1905     (xpart urcorner p - xpart center p) ++ (ypart urcorner p - ypart center p)
1906   )
1907 enddef;
1908 def withshadingstep (text t) =
1909   hide(mplib_shade_step := mplib_shade_step + 1 ;)
1910   withprescript "sh_step=" & decimal mplib_shade_step
1911   t
1912 enddef;
1913 let withfadestep = withshadingstep;
1914 def withshadingradius expr a =
1915   withprescript "sh_radius_a=" & decimal (xpart a)
1916   withprescript "sh_radius_b=" & decimal (ypart a)
1917 enddef;
1918 def withshadingorigin expr a =
1919   withprescript "sh_center_a=" & ddecimal a
1920   withprescript "sh_center_b=" & ddecimal a
1921 enddef;
1922 def withshadingvector expr a =
1923   withprescript "sh_center_a=" & ddecimal (point xpart a of mplib_shade_path)
1924   withprescript "sh_center_b=" & ddecimal (point ypart a of mplib_shade_path)
1925 enddef;
1926 def withshadingdirection expr a =
1927   withprescript "sh_center_a=" & ddecimal (point xpart a of boundingbox(mplib_shade_path))
1928   withprescript "sh_center_b=" & ddecimal (point ypart a of boundingbox(mplib_shade_path))
1929 enddef;
1930 def withshadingtransform expr a =
1931   withprescript "sh_transform=" & a
1932 enddef;
1933 def withshadingcenter expr a =
1934   withprescript "sh_center_a=" & ddecimal (
1935     center mplib_shade_path shifted (
1936       xpart a * xpart (lrcorner mplib_shade_path - llcorner mplib_shade_path)/2,
1937       ypart a * ypart (urcorner mplib_shade_path - lrcorner mplib_shade_path)/2
1938     )
1939   )
1940 enddef;
1941 def withshadingcenters (expr a, b) =
1942   withprescript "sh_center_a=" & ddecimal a
1943   withprescript "sh_center_b=" & ddecimal b
1944   withshadingtransform "no"
1945   withshadingfactor 1
1946 enddef;
1947 let withshadingpoints = withshadingcenters;

```

```

1948 def withshadingextend (expr a, b) =
1949   withprescript "sh_extend=" &
1950     if a: "true" else: "false" fi & " " &
1951     if b: "true" else: "false" fi
1952 enddef;
1953 let withfadeextend = withshadingextend;
1954 def withshadingdomain expr d =
1955   withprescript "sh_domain=" & ddecimal d
1956 enddef;
1957 def withshadingfactor expr f =
1958   withprescript "sh_factor=" & decimal f
1959 enddef;
1960 def withshadingfraction expr a =
1961   if mplib_shade_step > 0 :
1962     withprescript "sh_fraction_" & decimal mplib_shade_step & "=" & decimal a
1963   fi
1964 enddef;
1965 let withfadefraction = withshadingfraction;
1966 def withshadingcolors (expr a, b) =
1967   if mplib_shade_step > 0 :
1968     withprescript "sh_color=into"
1969     withprescript "sh_color_a_" & decimal mplib_shade_step & "=" & colordecimals a
1970     withprescript "sh_color_b_" & decimal mplib_shade_step & "=" & colordecimals b
1971   else :
1972     withprescript "sh_color=into"
1973     withprescript "sh_color_a=" & colordecimals a
1974     withprescript "sh_color_b=" & colordecimals b
1975   fi
1976 enddef;
1977 let withfadeopacity = withshadingcolors;
1978 def withshadingstroke expr a =
1979   withprescript "sh_stroking=" & a
1980 enddef;
1981 def withshadingmatrix expr s =
1982   withprescript "sh_matrix=" & s
1983 enddef;
1984 let withfadematrix = withshadingmatrix;
1985 def mpliblength primary t =
1986   runscript("return utf8.len[====[" & t & "]====]")
1987 enddef;
1988 def mplibsubstring expr p of t =
1989   runscript("return luamplib.unicodesubstring([====[" & t & "]====],"
1990     & decimal xpart p & ","
1991     & decimal ypart p & ")")
1992 enddef;
1993 def mplibuclength primary t =
1994   runscript("return #luamplib.getunicodegraphemes[====[" & t & "]====]")
1995 enddef;
1996 def mplibucsubstring expr p of t =

```

```

1997  runscript("return luamplib.unicodesubstring(====[" & t & "]====),"
1998    & decimal xpart p & ","
1999    & decimal ypart p & ",true)")
2000  enddef;
2001  ]],
2002  legacyverbatimtex = [[
2003  def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
2004  def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
2005  let VerbatimTeX = specialVerbatimTeX;
2006  extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
2007    "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
2008  extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
2009    "runscript(" &ditto&
2010    "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
2011    "luamplib.in_the_fig=false" &ditto& ");";
2012  ]],
2013  texttextlabel = [[
2014  let luampliboriginalinfont = infont;
2015  primarydef s infont f =
2016    if (s < char 32)
2017      or (s = char 35) % #
2018      or (s = char 36) % $
2019      or (s = char 37) % %
2020      or (s = char 38) % &
2021      or (s = char 92) % \
2022      or (s = char 94) % ^
2023      or (s = char 95) % _
2024      or (s = char 123) % {
2025      or (s = char 125) % }
2026      or (s = char 126) % ~
2027      or (s = char 127) :
2028      s luampliboriginalinfont f
2029    else :
2030      rawtexttext(s)
2031    fi
2032  enddef;
2033  def fontsize expr f =
2034    begingroup
2035    save size; numeric size;
2036    size := mplibdimen("1em");
2037    if size = 0: 10pt else: size fi
2038  endgroup
2039  enddef;
2040  ]],
2041  }
2042

```

process_mplibcode

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

```

2043 luamplib.verbatiminput = false
2044 luamplib.everymplib      = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2045 luamplib.everyendmplib  = setmetatable({ ["" ] = "" },{ __index = function(t) return t[""] end })
2046 function luamplib.process_mplibcode (data, instancename)
2047   texboxes.localid = 4096

```

This is needed for legacy behavior

```

2048 if luamplib.legacyverbatim then
2049   luamplib.figid, tex_code_pre_mplib = 1, {}
2050 end
2051 local everymplib      = luamplib.everymplib[instancename]
2052 local everyendmplib  = luamplib.everyendmplib[instancename]
2053 data = format("\n%s\n%s\n%s\n",everymplib, data, everyendmplib)
2054 :gsub("\r","\n")

```

These five lines are needed for mplibverbatim mode.

```

2055 if luamplib.verbatiminput then
2056   data = data:gsub("\mpcolor%+(-%b{})", "mplibcolor(\"%1\")")
2057   :gsub("\mpdim%+{%b{}}", "mplibdimen(\"%1\")")
2058   :gsub("\mpdim%+(\%a+)", "mplibdimen(\"%1\")")
2059   :gsub(btex_etex, "btex %1 etex ")
2060   :gsub(verbatimetex_etex, "verbatimetex %1 etex;")
2061 else

```

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

```

2062   local t = { } -- to store btex, verbatimetex, string
2063   data = data:gsub(btex_etex, function(str)
2064     t[#t+1] = str
2065     return format("btex \unexpanded{!l!u!a!%s!m!p!l!} etex ", #t) -- space
2066   end)
2067   :gsub(verbatimetex_etex, function(str)
2068     t[#t+1] = str
2069     return format("verbatimetex \unexpanded{!l!u!a!%s!m!p!l!} etex;", #t) -- semicolon
2070   end)
2071   :gsub("(.-)"', function(str)
2072     t[#t+1] = str
2073     return format("\unexpanded{!l!u!a!%s!m!p!l!}"', #t)
2074   end)
2075   :gsub("\%%", "\0PerCent\0")
2076   :gsub("%%.-\n", "\n")
2077   :gsub("%zPerCent%z", "\%")
2078   run_tex_code(format("\mplibtmptoks\expandafter{\expanded{%s}}", data))
2079   data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

2080   :gsub("##", "#")
2081   :gsub("!l!u!a!(%d+)!m!p!l!", function(str) return t[tonumber(str)] or str end)
2082 end

```

```

2083 process(data, instancename)
2084 end
2085

```

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

```

2086 local figcontents = { post = { } }
2087 local function put2output(a,...)
2088   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
2089 end
2090 local function pdf_startfigure(n,llx,lly,urx,ury)
2091   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury)
2092 end
2093 local function pdf_stopfigure()
2094   put2output("\mplibstoptoPDF")
2095 end

```

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

```

2096 local function pdf_literalcode (...)
2097   put2output{ -2, (format(...) :gsub(decimals,rmzeros)) }
2098 end
2099 local start_pdf_code = pdfmode
2100   and function() pdf_literalcode"q" end
2101   or function() put2output"\special{pdf:bcontent}" end
2102 local stop_pdf_code = pdfmode
2103   and function() pdf_literalcode"Q" end
2104   or function() put2output"\special{pdf:econtent}" end
2105

```

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

```

2106 local function put_tex_boxes (object,prescript)
2107   local box = prescript.mplibtexboxid:explode":"
2108   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
2109   if n and tw and th then
2110     local op = object.path
2111     local first, second, fourth = op[1], op[2], op[4]
2112     local tx, ty = first.x_coord, first.y_coord
2113     local sx, rx, ry, sy = 1, 0, 0, 1
2114     if tw ~= 0 then
2115       sx = (second.x_coord - tx)/tw
2116       rx = (second.y_coord - ty)/tw
2117       if sx == 0 then sx = 0.00001 end
2118     end
2119     if th ~= 0 then
2120       sy = (fourth.y_coord - ty)/th
2121       ry = (fourth.x_coord - tx)/th
2122       if sy == 0 then sy = 0.00001 end
2123     end
2124

```

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

```

2125 local matrix = format("%f %f %f %f", sx, rx, ry, sy) :gsub(decimals,rmzeros)
2126 put2output("\mplibputtextbox{%i}{%f}{%f}{%s}", n, tx, ty, matrix)
2127 end
2128 end
2129

```

Colors

```

2130 local do_preobj_CR
2131 do
2132 local prev_override_color
2133 function do_preobj_CR(object,prescript)
2134 if object.postscript == "collect" then return end
2135 local override = prescript and prescript.mpliboverridecolor
2136 if override then
2137 if pdfmode or override:find" cs " then
2138 pdf_literalcode(override)
2139 override = nil
2140 else
2141 put2output("\special{%s}",override)
2142 prev_override_color = override
2143 end
2144 else
2145 local cs = object.color
2146 if cs and #cs > 0 then
2147 pdf_literalcode(luamplib.colorconverter(cs))
2148 prev_override_color = nil
2149 elseif not pdfmode then
2150 override = prev_override_color
2151 if override then
2152 put2output("\special{%s}",override)
2153 end
2154 end
2155 end
2156 return override
2157 end
2158 end
2159

```

For transparency, shading, fading, and pattern

```

2160 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
2161 local pdfobjs, pdfetcs = {}, {}
2162 pdfetcs.pgfbxtgs = "pgf@sys@addpdfresource@extgs@plain"
2163 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
2164 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
2165 local update_pdfobjs
2166 if pdfmode then
2167 function update_pdfobjs (os, stream)
2168 local key = os
2169 if stream then key = key..stream end
2170 local on = key and pdfobjs[key]

```

```

2171     if on then
2172         return on,false
2173     end
2174     if stream then
2175         on = pdf.immediateobj("stream",stream,os)
2176     elseif os then
2177         on = pdf.immediateobj(os)
2178     else
2179         on = pdf.reserveobj()
2180     end
2181     if key then
2182         pdfobjs[key] = on
2183     end
2184     return on,true
2185 end
2186 else
2187     function update_pdfobjs (os, stream, hex)
2188         local key = os
2189         if stream then key = key..stream end
2190         local on = key and pdfobjs[key]
2191         if on then
2192             return on,false
2193         end
2194         on = pdfetcs.cnt or 1
2195         if stream then
2196             if hex then
2197                 texsprintf(format("\\special{pdf:stream @mplibpdfobj%s <%s> <<%s>>}",on,stream,os))
2198             else
2199                 texsprintf(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
2200             end
2201         elseif os then
2202             texsprintf(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
2203         else
2204             texsprintf(format("\\special{pdf:obj @mplibpdfobj%s <<>>}",on))
2205         end
2206         pdfetcs.cnt = on + 1
2207         if key then
2208             pdfobjs[key] = on
2209         end
2210         return on,true
2211     end
2212 end
2213 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
2214 if pdfmode then
2215     pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
2216     local getpagers = pdfetcs.getpagers
2217     local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
2218     local initialize_resources = function (name)
2219         local tabname = format("%s_res",name)

```

```

2220 pdfetcs[tabname] = { }
2221 if luatexbase.callbacktypes.finish_pdffile then -- ltluatex
2222   local obj = pdf.reserveobj()
2223   setpagemeres(format("%s/%s %i 0 R", getpagemeres() or "", name, obj))
2224   luatexbase.add_to_callback("finish_pdffile", function()
2225     pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
2226   end,
2227   format("luamplib.%s.finish_pdffile",name))
2228   end
2229 end
2230 pdfetcs.fallback_update_resources = function (name, res)
2231   local tabname = format("%s_res",name)
2232   if not pdfetcs[tabname] then
2233     initialize_resources(name)
2234   end
2235   if luatexbase.callbacktypes.finish_pdffile then
2236     local t = pdfetcs[tabname]
2237     t[#t+1] = res
2238   else
2239     local tpr, n = getpagemeres() or "", 0
2240     tpr, n = tpr:gsub(format("/%s<<",name), "%1"..res)
2241     if n == 0 then
2242       tpr = format("%s/%s<<%s>>", tpr, name, res)
2243     end
2244     setpagemeres(tpr)
2245   end
2246 end
2247 else
2248   texpstr {
2249     "\\luamplibatfirstshipout{",
2250     "\\special{pdf:obj @MPLibTr<<>>}",
2251     "\\special{pdf:obj @MPLibSh<<>>}",
2252     "\\special{pdf:obj @MPLibCS<<>>}",
2253     "\\special{pdf:obj @MPLibPt<<>>}}",
2254   }
2255 pdfetcs.fallback_update_resources = function (name,res,obj)
2256   texpstr{"\\special{pdf:put ", obj, " <<", res, ">>}" }
2257   local tabname = format("%s_res",name)
2258   if not pdfetcs[tabname] then
2259     texpstr{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}}"}
2260     pdfetcs[tabname] = { }
2261   end
2262   tableinsert(pdfetcs[tabname], res)
2263 end
2264 end
2265

```

Transparency

```

2266 local function add_extgs_resources (on, new)

```

```

2267 local key = format("MPLibTr%s", on)
2268 if new then
2269     local val = format(pdfetcs.resfmt, on)
2270     if pdfmanagement then
2271         texsprint {
2272             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
2273         }
2274     else
2275         local tr = format("/%s %s", key, val)
2276         if is_defined(pdfetcs.pgfextgs) then
2277             texsprint { "\\csname ", pdfetcs.pgfextgs, "\\endcsname{" , tr, "}" }
2278         elseif is_defined"TRP@list" then
2279             texsprint(catat11,{
2280                 [[\if@files\immediate\write\@auxout{]],
2281                 [[\string\g@addto@macro\string\TRP@list{]],
2282                 tr,
2283                 [[}}\fi]],
2284             })
2285             if not get_macro"TRP@list":find(tr) then
2286                 texsprint(catat11,[[\global\TRP@reruntrue]])
2287             end
2288         else
2289             pdfetcs.fallback_update_resources("ExtGState",tr,"@MPLibTr")
2290         end
2291     end
2292 end
2293 return key
2294 end
2295
2296 local do_preobj_TR
2297 do
2298     local transparency_modes = {
2299         [0] = "Normal",
2300         "Normal",      "Multiply",      "Screen",      "Overlay",
2301         "SoftLight",   "HardLight",   "ColorDodge", "ColorBurn",
2302         "Darken",      "Lighten",     "Difference",  "Exclusion",
2303         "Hue",          "Saturation",  "Color",      "Luminosity",
2304         "Compatible",
2305         normal        = "Normal",    multiply       = "Multiply",   screen        = "Screen",
2306         overlay       = "Overlay",    softlight     = "SoftLight",  hardlight     = "HardLight",
2307         colordodge    = "ColorDodge",  colorburn     = "ColorBurn",  darken        = "Darken",
2308         lighten       = "Lighten",     difference    = "Difference", exclusion     = "Exclusion",
2309         hue           = "Hue",         saturation    = "Saturation", color          = "Color",
2310         luminosity    = "Luminosity",  compatible    = "Compatible",
2311     }
2312 function do_preobj_TR(object,prescript)
2313     if object.postscript == "collect" then return end
2314     local opa = prescript and prescript.tr_transparency
2315     if not opa then return end

```

```

2316
2317 local key, on, os, new
2318 local mode = prescript.tr_alternative or 1
2319 mode = transparency_modes[tonumber(mode) or mode:lower()]
2320 if not mode then
2321     mode = prescript.tr_alternative
2322     warn("unsupported blend mode: '%s'", mode)
2323 end
2324 opaq = opaq:explode":""
2325 for i,v in ipairs(opaq) do
2326     opaq[i] = format("%.3f", v) :gsub(decimals,rmzeros)
2327 end
2328 for i,v in ipairs[ {mode,opaq[1],opaq[2] or opaq[1]},{ "Normal",1,1} ] do
2329     os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[3])
2330     on, new = update_pdfobjs(os)
2331     key = add_extgs_resources(on,new)
2332     if i == 1 then
2333         pdf_literalcode("/%s gs",key)
2334     else
2335         return format("/%s gs",key)
2336     end
2337 end
2338 end
2339 end
2340

```

Shading with *metafun* format.

```

2341 local function add_shading_resources (on, new)
2342     if new then
2343         local key, val = format("MPLibSh%s", on), format(pdfetcs.resfmt, on)
2344         if pdfmanagement then
2345             texsprint {
2346                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{", val, "}"
2347             }
2348         else
2349             local res = format("/%s %s", key, val)
2350             pdfetcs.fallback_update_resources("Shading",res,"@MPLibSh")
2351         end
2352     end
2353 end
2354 local function sh_pdfpageresources(shtype,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2355     for _,v in ipairs{ca,cb} do
2356         for i,vv in ipairs(v) do
2357             for ii,vvv in ipairs(vv) do
2358                 v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
2359             end
2360         end
2361     end
2362     local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"

```

```

2363 if steps > 1 then
2364   local list,bounds,encode = { },{ },{ }
2365   for i=1,steps do
2366     if i < steps then
2367       bounds[i] = format("%.3f", fractions[i] or 1)
2368     end
2369     encode[2*i-1] = 0
2370     encode[2*i]   = 1
2371     os = fun2fmt:format(domain,tableconcat(ca[i],' '),tableconcat(cb[i],' '))
2372       :gsub(decimals,rmzeros)
2373     list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
2374   end
2375   os = tableconcat {
2376     "<</FunctionType 3",
2377     format("/Bounds[%s]",   tableconcat(bounds,' ')),
2378     format("/Encode[%s]",   tableconcat(encode,' ')),
2379     format("/Functions[%s]", tableconcat(list, ' ')),
2380     format("/Domain[%s]>>", domain),
2381   } :gsub(decimals,rmzeros)
2382 else
2383   os = fun2fmt:format(domain,tableconcat(ca[1],' '),tableconcat(cb[1],' '))
2384     :gsub(decimals,rmzeros)
2385 end
2386 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
2387 os = tableconcat {
2388   format("<</ShadingType %i", shtype),
2389   format("/ColorSpace %s",   colorspace),
2390   format("/Function %s",     objref),
2391   format("/Coords[%s]",     coordinates),
2392   format("/Extend[%s]/AntiAlias true>>", extend or "true true")
2393 } :gsub(decimals,rmzeros)
2394 local on, new = update_pdfobjs(os)
2395 add_shading_resources(on, new)
2396 return on
2397 end
2398
2399 local function get_mp_matrix (matrix)
2400   process("&mplibtransformmatrix(%s);"):format(matrix), "&mplibtransformmatrix")
2401   return luamplib.transformmatrix
2402 end
2403
2404 local do_preobj_SH
2405 do
2406   pdfetcs.clrprocs = setmetatable({ }, { __index = function(t,names)
2407     run_tex_code({
2408       [[\color_model_new:nnn]],
2409       format("&mplibcolorspace_%s", names:gsub(",","_")),
2410       format("&DeviceN}{names={%s}}", names),
2411       [[\mplibmptoks\expanded{\pdf_object_ref_last:}]],

```

```

2412     }, ccexplat)
2413     local colorspace = texgettoks"mplibtmptoks"
2414     t[names] = colorspace
2415     return colorspace
2416 end })
2417 local function color_normalize(ca,cb)
2418     if #cb == 1 then
2419         if #ca == 4 then
2420             cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
2421         else -- #ca = 3
2422             cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
2423         end
2424     elseif #cb == 3 then -- #ca == 4 : rgb to cmyk
2425         local c, m, y = 1-cb[1], 1-cb[2], 1-cb[3]
2426         local k = math.min(c, m, y)
2427         if k == 1 then
2428             c, m, y = 0, 0, 0
2429         else
2430             c, m, y = (c-k)/(1-k), (m-k)/(1-k), (y-k)/(1-k)
2431         end
2432         cb[1], cb[2], cb[3], cb[4] = c, m, y, k
2433     end
2434 end
2435 local function write_mesh_objs (shtype, colorspace, stream, devicen, perrow)
2436     local cc = colorspace == "/DeviceCMYK" and 4
2437         or colorspace == "/DeviceRGB" and 3
2438         or devicen or 1
2439     local dict = format(
2440         "/ShadingType %d/%s/BitsPerCoordinate 16/BitsPerComponent 8/ColorSpace %s/Decode[0 1 0 1%s]",
2441         shtype,
2442         perrow and format("VerticesPerRow %d", perrow) or "BitsPerFlag 8",
2443         colorspace,
2444         (" 0 1"):rep(cc))
2445     local on, new
2446     if pdfmode then
2447         on, new = update_pdfobjs(dict, stream)
2448     else
2449         stream = format("%02X"):rep(stream:len()), stream:byte(1,stream:len())
2450         on, new = update_pdfobjs(dict, stream, true) -- hex is true
2451     end
2452     add_shading_resources(on, new)
2453     return on
2454 end
2455 local function colors_ff (colors)
2456     local t, devicen = { }
2457     for i,v in ipairs(colors) do
2458         t[i] = { }
2459         for _,vv in ipairs(v) do
2460             local tt = tableconcat(vv," "):explode()

```

```

2461     for _,vvv in ipairs(tt) do
2462         tableinsert(t[i], string.char(math.floor(vvv * 0xFF)))
2463     end
2464     devicen = #tt
2465 end
2466 end
2467 return t, devicen
2468 end
2469 local function coords_ffff (coords)
2470     local Xt, Yt = { }, { }
2471     for i,v in ipairs(coords) do
2472         for ii,vv in ipairs(v) do
2473             local t = ii % 2 == 1 and Xt or Yt
2474             t[#t+1] = tonumber(vv)
2475         end
2476     end
2477     local xmin, xmax = math.min(tableunpack(Xt)), math.max(tableunpack(Xt))
2478     local ymin, ymax = math.min(tableunpack(Yt)), math.max(tableunpack(Yt))
2479     local wd, ht = xmax - xmin, ymax - ymin
2480     for i,v in ipairs(coords) do
2481         for ii,vv in ipairs(v) do
2482             local min = ii % 2 == 1 and xmin or ymin
2483             local dim = ii % 2 == 1 and wd or ht
2484             coords[i][ii] = string.pack(">H", math.floor((vv - min)/dim * 0xFFFF ))
2485         end
2486     end
2487     return coords, xmin, ymin, wd, ht
2488 end
2489 local function do_shading_lattice_mesh (object, prescript, colorspace, ca, cb)
2490     local perrow = prescript.sh_lattice_perrow or 2
2491     local steps = tonumber(prescript.sh_step) or 0
2492     local coords, colors = { }, { }
2493     if steps < 4 then
2494         local path = object.path
2495         for _,i in ipairs{1,2,4,3} do
2496             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2497         end
2498         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}}, {{1,1,0}} }
2499         colorspace = "/DeviceRGB"
2500     else
2501         local t = prescript.sh_lattice_data:explode()
2502         for i = 1, #t, 2 do
2503             coords[#coords+1] = { t[i], t[i+1] }
2504         end
2505         for i = 1, steps do
2506             colors[#colors+1] = { ca[i] }
2507         end
2508     end
2509     if #coords % perrow ~= 0 then err"vertices_per_row mismatches lattice_coords" end

```

```

2510
2511 local stream, xmin, ymin, wd, ht, devicen = { }
2512 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2513 colors, devicen = colors_ff(colors)
2514 for i = 1, #coords do
2515     stream[#stream+1] = tableconcat(coords[i])
2516     if not colors[i] then err"colorspace mismatch?" end
2517     stream[#stream+1] = tableconcat(colors[i])
2518 end
2519
2520 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2521 local on = write_mesh_objs (5, colorspace, tableconcat(stream), devicen, perrow)
2522 return on, matrix
2523 end
2524 local function do_shading_triangle_mesh (object, prescript, colorspace, ca, cb)
2525     local steps = tonumber(prescript.sh_step) or 0
2526     local coords, colors, edges = { }, { }, { }
2527     if steps < 3 then
2528         local path = object.path
2529         for i = 1, 3 do
2530             coords[#coords+1] = { path[i].x_coord, path[i].y_coord }
2531         end
2532         colors = { {{1,0,0}}, {{0,1,0}}, {{0,0,1}} }
2533         edges = { 0, 0, 0 }
2534         colorspace = "/DeviceRGB"
2535     else
2536         for i = 1, steps do
2537             local t = prescript["sh_triangle_vertex_" .. i]:explode()
2538             if #t == 2 then
2539                 coords[#coords+1] = { t[1], t[2] }
2540             elseif #t == 6 then
2541                 coords[#coords+1] = { t[1], t[2] }
2542                 coords[#coords+1] = { t[3], t[4] }
2543                 coords[#coords+1] = { t[5], t[6] }
2544             end
2545             colors[#colors+1] = { ca[i] }
2546             edges[#edges+1] = tonumber(prescript["sh_triangle_edge_" .. i])
2547         end
2548     end
2549
2550 local stream, xmin, ymin, wd, ht, devicen = { }
2551 coords, xmin, ymin, wd, ht = coords_ffff(coords)
2552 colors, devicen = colors_ff(colors)
2553 for i = 1, #coords do
2554     stream[#stream+1] = string.char(edges[i])
2555     stream[#stream+1] = tableconcat(coords[i])
2556     if not colors[i] then err"colorspace mismatch?" end
2557     stream[#stream+1] = tableconcat(colors[i])
2558 end

```

```

2559
2560     local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2561     local on = write_mesh_objs (4, colorspace, tableconcat(stream), devicen)
2562     return on, matrix
2563 end
2564 local function do_shading_coons_patch (object, prescript, colorspace, ca, cb)
2565     local tensor = prescript.sh_type == "tensor"
2566     local steps = tonumber(prescript.sh_step) or 0
2567     local coords, colors, edges = { }, { }, { }
2568     if steps < 4 then
2569         local path, t = object.path, { }
2570         for i = 1, 4 do
2571             t[#t+1] = path[i].x_coord
2572             t[#t+1] = path[i].y_coord
2573             t[#t+1] = path[i].right_x
2574             t[#t+1] = path[i].right_y
2575             local j = i == 4 and 1 or i+1
2576             t[#t+1] = path[j].left_x
2577             t[#t+1] = path[j].left_y
2578         end
2579         if tensor then
2580             for _,v in ipairs( prescript.sh_tensor_path:explode() ) do
2581                 t[#t+1] = v
2582             end
2583         end
2584         coords = { t }
2585         colors = {{ {1,0,0}, {0,1,0}, {0,0,1}, {1,1,0} }}
2586         edges = { 0 }
2587         colorspace = "/DeviceRGB"
2588     else
2589         for i = 1, steps do
2590             local edge = tonumber(prescript["sh_coons_edge_".i])
2591             if edge then
2592                 edges[#edges+1] = edge
2593                 coords[#coords+1] = prescript["sh_coons_path_".i]:explode()
2594                 if edge == 0 then
2595                     colors[#colors+1] = { ca[i], cb[i], ca[i+1], cb[i+1] }
2596                 else
2597                     colors[#colors+1] = { ca[i], cb[i] }
2598                 end
2599             end
2600         end
2601     end
2602
2603     local stream, xmin, ymin, wd, ht, devicen = { }
2604     coords, xmin, ymin, wd, ht = coords_ffff(coords)
2605     colors, devicen = colors_ff(colors)
2606     for i = 1, #coords do
2607         stream[#stream+1] = string.char(edges[i])

```

```

2608     stream[#stream+1] = tableconcat(coords[i])
2609     if not colors[i] then err"colorspace mismatch?" end
2610     stream[#stream+1] = tableconcat(colors[i])
2611 end
2612
2613 local matrix = format("%f 0 0 %f %f %f", wd, ht, xmin, ymin) :gsub(decimals,rmzeros)
2614 local on = write_mesh_objjs (tensor and 7 or 6, colorspace, tableconcat(stream), devicen)
2615 return on, matrix
2616 end
2617 function do_preobj_SH(object, prescript)
2618     local shade_no
2619     local sh_type = prescript and prescript.sh_type
2620     if not sh_type then return end
2621
2622     local domain = prescript.sh_domain or "0 1"
2623     local centera = (prescript.sh_center_a or "0 0"):explode()
2624     local centerb = (prescript.sh_center_b or "0 0"):explode()
2625     local transform = prescript.sh_transform == "yes"
2626     local sx,sy,sr,dx,dy = 1,1,1,0,0
2627     if transform then
2628         local first = (prescript.sh_first or "0 0"):explode()
2629         local setx = (prescript.sh_set_x or "0 0"):explode()
2630         local sety = (prescript.sh_set_y or "0 0"):explode()
2631         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
2632         if x ~= 0 and y ~= 0 then
2633             local path = object.path
2634             local path1x = path[1].x_coord
2635             local path1y = path[1].y_coord
2636             local path2x = path[x].x_coord
2637             local path2y = path[y].y_coord
2638             local dxa = path2x - path1x
2639             local dya = path2y - path1y
2640             local dxb = setx[2] - first[1]
2641             local dyb = sety[2] - first[2]
2642             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
2643                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
2644                 sy = dya / dyb ; if sy < 0 then sy = - sy end
2645                 sr = math.sqrt(sx^2 + sy^2)
2646                 dx = path1x - sx*first[1]
2647                 dy = path1y - sy*first[2]
2648             end
2649         end
2650     end
2651     local ca, cb, colorspace, steps, fractions
2652     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:"" }
2653     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:"" }
2654     steps = tonumber(prescript.sh_step) or 1
2655     if steps > 1 then
2656         fractions = { prescript.sh_fraction_1 or 0 }

```

```

2657     for i=2,steps do
2658         fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
2659         ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
2660         cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
2661     end
2662 end
2663 if prescript.mplib_spotcolor then
2664     ca, cb = { }, { }
2665     local names, pos, objref = { }, -1, ""
2666     local script = object.prescript:explode"\13+"
2667     for i=#script,1,-1 do
2668         local name, value
2669         if script[i]:find"mplib_spotcolor" then
2670             local t = script[i]:explode"="[2]:explode":"
2671             value, objref, name = t[1], t[2], t[3]
2672         elseif script[i]:find"mplib_texcolor" then
2673             local t = script[i]:explode"="[2]:explode"!"
2674             name, value = t[1], (t[2] or 100)/100
2675         end
2676         if name and value then
2677             if not names[name] then
2678                 pos = pos+1
2679                 names[name] = pos
2680                 names[#names+1] = name
2681             end
2682             local t = { }
2683             for j=1,names[name] do t[#t+1] = 0 end
2684             t[#t+1] = value
2685             tableinsert(#ca == #cb and ca or cb, t)
2686         end
2687     end
2688     for _,t in ipairs{ca,cb} do
2689         for _,tt in ipairs(t) do
2690             for i=1,#names-#tt do tt[#tt+1] = 0 end
2691         end
2692     end
2693     if #names == 1 then
2694         colorspace = objref
2695     else
2696         colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
2697     end
2698 else
2699     local model = 0
2700     for _,t in ipairs{ca,cb} do
2701         for _,tt in ipairs(t) do
2702             model = model > #tt and model or #tt
2703         end
2704     end
2705     for _,t in ipairs{ca,cb} do

```

```

2706     for _,tt in ipairs(t) do
2707         if #tt < model then
2708             color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
2709         end
2710     end
2711 end
2712 colorspace = model == 4 and "/DeviceCMYK"
2713             or model == 3 and "/DeviceRGB"
2714             or model == 1 and "/DeviceGray"
2715             or err"unknown color model"
2716 end
2717 local extend = prescript.sh_extend
2718 local mesh_matrix
2719 if sh_type == "linear" then
2720     local coordinates = format("%f %f %f %f",
2721         dx + sx*centera[1], dy + sy*centera[2],
2722         dx + sx*centerb[1], dy + sy*centerb[2])
2723     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2724 elseif sh_type == "circular" then
2725     local factor = prescript.sh_factor or 1
2726     local radiusa = factor * prescript.sh_radius_a
2727     local radiusb = factor * prescript.sh_radius_b
2728     local coordinates = format("%f %f %f %f %f %f",
2729         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
2730         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
2731     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions,extend)
2732 elseif sh_type == "coons" or sh_type == "tensor" then
2733     shade_no, mesh_matrix = do_shading_coons_patch(object, prescript, colorspace, ca, cb)
2734 elseif sh_type == "triangle" then
2735     shade_no, mesh_matrix = do_shading_triangle_mesh(object, prescript, colorspace, ca, cb)
2736 elseif sh_type == "lattice" then
2737     shade_no, mesh_matrix = do_shading_lattice_mesh(object, prescript, colorspace, ca, cb)
2738 else
2739     err"unknown shading type"
2740 end
2741
2742 local matrix = prescript.sh_matrix
2743 if matrix and matrix:find"%a" then
2744     local t = get_mp_matrix(matrix)
2745     matrix = format("%f %f %f %f %f %f", tableunpack(t)) :gsub(decimals,rmzeros)
2746 end
2747 if mesh_matrix and matrix then
2748     local a, b = mesh_matrix:explode(), matrix:explode()
2749     matrix = format("%f %f %f %f %f %f",
2750         a[1]*b[1]+a[2]*b[3], a[1]*b[2]+a[2]*b[4], a[3]*b[1]+a[4]*b[3], a[3]*b[2]+a[4]*b[4],
2751         a[5]+b[5], a[6]+b[6]) :gsub(decimals,rmzeros)
2752 end
2753
2754 return shade_no, prescript.sh_stroking == "yes", matrix or mesh_matrix

```

```

2755 end
2756 end
2757

```

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

```

2758 local function add_pattern_resources (key, val)
2759   if pdfmanagement then
2760     texsprint {
2761       "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{" , val, "}"
2762     }
2763   else
2764     local res = format("/%s %s", key, val)
2765     if is_defined(pdfetcs.pgfpattern) then
2766       texsprint { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{" , res, "}" }
2767     else
2768       pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2769     end
2770   end
2771 end
2772 if pdfmode then
2773   function luamplib.dolatelua (on, os, matrix)
2774     local h, v = pdf.getpos()
2775     local t = matrix and matrix:explode() or {1,0,0,1,0,0}
2776     matrix = format("%f %f %f %f %f %f", t[1], t[2], t[3], t[4], t[5]+h/factor, t[6]+v/factor)
2777     :gsub(decimals,rmzeros)
2778     pdf.obj(on, format("<<%s/Matrix[%s]>>", os, matrix))
2779     pdf.refobj(on)
2780   end
2781 else
2782   pdfetcs.shadingpatterns = { }
2783   pdfetcs.shadingpatterninit_r, pdfetcs.shadingpatterninit_w = true, true
2784   function luamplib.dolatelua (on, kind, xobj)
2785     local h, v
2786     local t = pdfetcs.shadingpatterns[on] or { }
2787     local shift = kind == "group" and pdfetcs.tr_group.shifts[xobj]
2788       or kind == "pattern" and pdfetcs.patterns[xobj].shifts
2789     if shift then
2790       h, v = -shift[1], -shift[2] -- engine bug in dvi mode?
2791     else
2792       h, v = pdf.getpos()
2793       h = format("%f", h/factor) :gsub(decimals,rmzeros)
2794       v = format("%f", v/factor) :gsub(decimals,rmzeros)
2795     end
2796     if tonumber(h) ~= tonumber(t[1]) or tonumber(v) ~= tonumber(t[2]) then
2797       warn"Rerun to get correct shading pattern"
2798     end
2799     local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2800     local init = pdfetcs.shadingpatterninit_w
2801     if init then pdfetcs.shadingpatterninit_w = nil end

```

```

2802 local f = iopen(name, init and "w" or "a")
2803 if f then
2804     f:write((" %s %s %s\n"):format(on, h, v))
2805     f:close()
2806 else
2807     err"cannot write a file. check the cache dir path"
2808 end
2809 end
2810 end
2811 local function do_preobj_shading (object, prescript)
2812 if not prescript or not prescript.sh_operand_type then return end
2813 local on,_,matrix = do_preobj_SH(object, prescript)
2814 local os = format("/PatternType 2/Shading %s", format(pdfetcs.resfmt, on))
2815 matrix = matrix or "1 0 0 1 0 0"
2816 if prescript.sh_in_xobj == "yes" then
2817     on = update_pdfobjs(("<<%s/Matrix[%s]>>"):format(os, matrix))
2818     goto skip_latelua
2819 end
2820 on = update_pdfobjs()
2821 if pdfmode then
2822     put2output(tableconcat{"\\latelua{luamplib.dolatelua(", on, "[", os, "], [", matrix, "]}")})
2823 else
2824     local xobj = is_defined"mplibgroupname" and {"group", get_macro"mplibgroupname"}
2825                 or is_defined"mplibpatternname" and {"pattern", get_macro"mplibpatternname"}
2826     local init = pdfetcs.shadingpatterninit_r
2827     if init then
2828         pdfetcs.shadingpatterninit_r = nil
2829         local name = format("%s/%s_shadingpatterns.aux", cachedir or outputdir(), tex.jobname)
2830         local f = iopen(name)
2831         if f then
2832             for line in f:lines() do
2833                 local t = line:explode()
2834                 pdfetcs.shadingpatterns[ tonumber(t[1]) ] = { t[2], t[3] }
2835             end
2836             f:close()
2837         end
2838     end

```

This seems to be needed for proper functioning:

```

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

2839 local t = pdfetcs.shadingpatterns[on] or { 0, 0 }
2840 local mt = matrix:explode()
2841 matrix = format("%s %s %s %s %s %s", mt[1], mt[2], mt[3], mt[4], mt[5]+t[1], mt[6]+t[2])
2842 texsprint{ "\\special{pdf:put ", format(pdfetcs.resfmt, on),
2843             format(" <<%s/Matrix[%s]>>", os, matrix) }
2844 put2output("\\latelua{ luamplib.dolatelua(%s,%s) }", on,
2845             xobj and ("'s',[[s]]"):format(xobj[1], xobj[2]))

```

```

2846 end
2847 ::skip_latelua::
2848 local key, val = format("MPLibPt%s", on), format(pdfetcs.resfmt, on)
2849 add_pattern_resources(key, val)
2850 pdf_literalcode("/Pattern cs/%s scn", key)

```

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

```

2851 prescript.sh_type = nil
2852 end
2853

```

Tiling Patterns

```

2854 pdfetcs.patterns = { }
2855 local function gather_resources (optres, ispattern)
2856 local t = { }
2857 if pdfmanagement then
2858 for _,v in ipairs {"ExtGState", "ColorSpace", "Pattern", "Shading"} do
2859 local mytoks
2860 run_tex_code ({
2861 "\mplibmptoks\expanded{",
2862 "\pdfdict_if_empty:nF{g__pdf_Core/Page/Resources/", v, "}",
2863 "{\pdfdict_use:n{g__pdf_Core/Page/Resources/", v, "}}", "}"},
2864 }, ccexplat)
2865 mytoks = texgettoks"mplibmptoks"
2866 if not pdfmode then
2867 mytoks = mytoks:gsub("\str_convert_pdfname:n%s*{(-)}", "%1") -- why not expanded?
2868 end
2869 mytoks = mytoks and mytoks:gsub("^%s*(.)%s*$", "%1")
2870 if mytoks and mytoks ~= "" then
2871 t[#t+1] = ("%s<<%s>>"):format(v, mytoks)
2872 end
2873 end
2874 elseif is_defined(pdfetcs.pgfextgs) then
2875 run_tex_code"\relax" -- flush tex.sprint queue
2876 if pdfmode then
2877 for k,v in pairs { ExtGState = "pgf@sys@pgf@resource@list@extgs",
2878 ColorSpace = "pgf@sys@pgf@resource@list@colorspaces",
2879 Pattern = "pgf@sys@pgf@resource@list@patterns", } do
2880 local res = (get_macro(v) or ""):gsub("^%s*(.)%s*$", "%1")
2881 if res ~= "" then
2882 t[#t+1] = ("%s<<%s>>"):format(k, res )
2883 end
2884 end
2885 else
2886 local abc = get_macro"pgfutil@abc" or ""
2887 for k,v in pairs { ExtGState = "@pgfextgs",
2888 ColorSpace = "@pgfcolorspaces",
2889 Pattern = "@pgfpatterns", } do
2890 local tt = { }
2891 for vv in abc:gmatch( v .. "%s*(%b<>)" ) do

```

```

2892         tt[#tt+1] = vv:match("^<<%s*(.)%s*>>$")
2893     end
2894     if #tt > 0 then
2895         t[#t+1] = ("%s<<%s>>"):format(k, tableconcat(tt) )
2896     end
2897 end
2898 end

```

We still have to deal with Shading resources.

```

2899     if luatexbase.callbacktypes.finish_pdffile then
2900         if pdfetcs.Shading_res then
2901             t[#t+1] = ("/Shading<<%s>>"):format( tableconcat(pdfetcs.Shading_res) )
2902         end
2903     else
2904         local res = pdfetcs.getpagerses()
2905         res = res and res:match"/Shading%s*%b<>"
2906         if res then
2907             t[#t+1] = res
2908         end
2909     end
2910 else
2911     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

```

2912         warn"transparent package is not fully functional without pdfmanagement code."
2913     end
2914     if luatexbase.callbacktypes.finish_pdffile then
2915         for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2916             local tt = pdfetcs[v.."_res"]
2917             if tt then
2918                 t[#t+1] = ("%s<<%s>>"):format(v, tableconcat(tt))
2919             end
2920         end
2921     else
2922         local res = pdfetcs.getpagerses()
2923         if res then
2924             t[#t+1] = res
2925         end
2926     end
2927 end
2928 local result = tableconcat(t)
2929 if optres ~= "" then
2930     for _,v in ipairs {"ExtGState","ColorSpace","Pattern","Shading"} do
2931         local res = optres:match("/"..v.."%s*%b<>")
2932         if res then
2933             if result:find("/"..v) then
2934                 res = res:match("<<(.)>>$")
2935                 result = result:gsub("/"..v.."%s*<<", "%1"..res, 1)
2936             else

```

```

2937         result = result .. res
2938     end
2939 end
2940 end
2941 end
2942 return result
2943 end
2944 function luamplib.registerpattern ( boxid, name, opts )
2945     local box = texgetbox(boxid)
2946     local wd = format("%.3f",box.width/factor)
2947     local hd = format("%.3f",(box.height+box.depth)/factor)
2948     info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2949     if opts.xstep == 0 then opts.xstep = nil end
2950     if opts.ystep == 0 then opts.ystep = nil end
2951     if opts.colored == nil then
2952         opts.colored = opts.coloured
2953         if opts.colored == nil then
2954             opts.colored = true
2955         end
2956     end
2957     if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2958     if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2959     if opts.matrix and opts.matrix:find"%a" then
2960         local t = get_mp_matrix(opts.matrix)
2961         opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2962         opts.xshift = opts.xshift or format("%f",t[5])
2963         opts.yshift = opts.yshift or format("%f",t[6])
2964     end
2965     local attr = {
2966         "/Type/Pattern",
2967         "/PatternType 1",
2968         format("/PaintType %i", opts.colored and 1 or 2),
2969         "/TilingType 2",
2970         format("/XStep %s", opts.xstep or wd),
2971         format("/YStep %s", opts.ystep or hd),
2972         format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2973     }
2974     local optres = opts.resources or ""
2975     optres = gather_resources(optres, true) -- tiling pattern plus masking glitches with acrobat
2976     local patterns = pdfetcs.patterns
2977     if pdfmode then
2978         if opts.bbox then
2979             attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2980         end
2981         attr = tableconcat(attr) :gsub(decimals,rmzeros)
2982         local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)
2983         patterns[name] = { id = index, colored = opts.colored }
2984     else
2985         local cnt = #patterns + 1

```

```

2986 local objname = "@mplibpattern" .. cnt
2987 local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2988 texsprintf {
2989     "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2990     "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2991     "\\hbox{\\unhbox ", boxid, "\\}\\luamplibatnextshipout{",
2992     "\\special{pdf:bcontent}",
2993     "\\special{pdf:bobj ", objname, " ", metric, "}",
2994     "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2995     "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2996     "\\special{pdf:put @resources <<", optres, ">>}",
2997     "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2998     "\\special{pdf:econtent}}",
2999 }
3000 patterns[cnt] = objname
3001 patterns[name] = { id = cnt, colored = opts.colored }
3002 patterns[name].shifts = { get_macro"MPllx", get_macro"MPlly" } -- for shading patterns above
3003 end
3004 end
3005
3006 local do_preobj_PAT
3007 do
3008 local function pattern_colorspace (cs)
3009 local on, new = update_pdfobjs(format("[/Pattern %s]", cs))
3010 if new then
3011 local key, val = format("MPlibCS%i",on), format(pdfetcs.resfmt,on)
3012 if pdfmanagement then
3013 texsprintf {
3014     "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{", val, "}"
3015 }
3016 else
3017 local res = format("/%s %s", key, val)
3018 if is_defined(pdfetcs.pgfcOLORSPACE) then
3019 texsprintf { "\\csname ", pdfetcs.pgfcOLORSPACE, "\\endcsname{", res, "}" }
3020 else
3021 pdfetcs.fallback_update_resources("ColorSpace",res,"@MPlibCS")
3022 end
3023 end
3024 end
3025 return on
3026 end
3027 function do_preobj_PAT(object, prescript)
3028 local name = prescript and prescript.mplibpattern
3029 if not name then return end
3030 local patterns = pdfetcs.patterns
3031 local patt = patterns[name]
3032 local index = patt and patt.id or err("cannot get pattern object '%s'", name)
3033 local key = format("MPlibPt%s",index)
3034 if patt.colored then

```

```

3035 pdf_literalcode("/Pattern cs /%s scn", key)
3036 else
3037 local color = prescript.mpliboverridecolor
3038 if not color then
3039 local t = object.color
3040 color = t and #t>0 and luamplib.colorconverter(t)
3041 end
3042 if not color then return end
3043 local cs
3044 if color:find" cs " or color:find"@pdf.obj" then
3045 if pdfmode then
3046 local name
3047 name, color = color:match"^(.-) cs (.-) sc"
3048 cs = format("%s 0 R", ltx.pdf.object_id( name:sub(2,-1) ))
3049 if cs == "0 0 R" then -- assumes colorspace.sty
3050 _, cs = get_spc_name_obj(name)
3051 end
3052 elseif color:find" cs " then
3053 local name
3054 name, color = color:match"^(.-) cs (.-) sc"
3055 run_tex_code({ "\\mplibmptoks\\expanded{{\\pdf_object_ref:n{" , name, "}}}" }, ccexplat)
3056 cs = texgettoks"mplibmptoks"
3057 else -- legacy: to be removed
3058 cs, color = color:match"pdf:bc (.-) %[(-)%]"
3059 end
3060 else
3061 local t = colorsplit(color)
3062 cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
3063 color = tableconcat(t, " ")
3064 end
3065 pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
3066 end
3067 if not patt.done then
3068 local val = pdfmode and format("%s 0 R", index) or patterns[index]
3069 add_pattern_resources(key, val)
3070 end
3071 patt.done = true
3072 end
3073 end
3074

```

Fading

```

3075 pdfetcs.fading = { }
3076 local function do_preobj_FADE (object, prescript)
3077 local fd_type = prescript and prescript.mplibfadetype
3078 local fd_stop = prescript and prescript.mplibfadestate
3079 if not fd_type then
3080 return fd_stop -- returns "stop" (if picture) or nil
3081 end

```

```

3082 local on, os, new
3083 if fd_type == "masking" then
3084   local mac = get_macro("luamplib.group"..prescript.mplibmaskname)
3085   on = mac:match(pdfmode and "%d+" or "{pdf:uxobj (.-)}")
3086   local bc = prescript.mplibmaskingbgcolor
3087   bc = bc and bc:gsub(":", " ")
3088   bc = bc and ("/BC[%s]"):format(bc):gsub(decimals,rmzeros) or ""
3089   os = format("<</SMask<</S/Luminosity/G %s%>>>>",
3090             pdfmode and format(pdfetcs.resfmt, on) or on, bc)
3091 else
3092   local bbox = prescript.mplibfadebbox:explode":""
3093   local dx, dy = -bbox[1], -bbox[2]
3094   local vec = prescript.mplibfadevector; vec = vec and vec:explode":""
3095   if not vec then
3096     if fd_type == "linear" then
3097       vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
3098     else
3099       local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
3100       vec = {centerx, centery, centerx, centery} -- center for both circles
3101     end
3102   end
3103   local coords = { vec[1], vec[2], vec[3], vec[4] }
3104   if fd_type == "linear" then
3105     coords = format("%f %f %f %f", tableunpack(coords))
3106   elseif fd_type == "circular" then
3107     local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
3108     local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode":""
3109     tableinsert(coords, 3, radius[1])
3110     tableinsert(coords, radius[2])
3111     coords = format("%f %f %f %f %f %f", tableunpack(coords))
3112   else
3113     err("unknown fading method '%s'", fd_type)
3114   end
3115   fd_type = fd_type == "linear" and 2 or 3
3116   local extend, steps, fractions = prescript.sh_extend, tonumber(prescript.sh_step) or 1
3117   local ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "1"):explode":"" }
3118   local cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "0"):explode":"" }
3119   if steps > 1 then
3120     fractions = { prescript.sh_fraction_1 or 0 }
3121     for i=2,steps do
3122       fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
3123       ca[i] = (prescript[format("sh_color_a_%i",i)] or "1"):explode":""
3124       cb[i] = (prescript[format("sh_color_b_%i",i)] or "0"):explode":""
3125     end
3126   end
3127   local matrix = prescript.sh_matrix or "1 0 0 1 0 0"
3128   matrix = matrix:find"%a" and get_mp_matrix(matrix) or matrix:explode()
3129   matrix[5] = matrix[5] + dx
3130   matrix[6] = matrix[6] + dy

```

```

3131 matrix = format("%f %f %f %f %f %f", tableunpack(matrix)) :gsub(decimals,rmzeros)
3132 on = sh_pdfpageresources(fd_type,"0 1","/DeviceGray",ca,cb,coords,steps,fractions,extend)
3133 os = format("<</PatternType 2/Shading %s/Matrix[%s]>>", format(pdfetcs.resfmt, on), matrix)
3134 on = update_pdfobjs(os)
3135 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
3136 local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
3137 :gsub(decimals,rmzeros)
3138 os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
3139 on = update_pdfobjs(os)
3140 local resources = format(pdfetcs.resfmt, on)
3141 on = update_pdfobjs"<</S/Transparency/CS/DeviceGray>>"
3142 local attr = tableconcat{
3143   "/Subtype/Form",
3144   "/BBox[" , bbox, "]",
3145   "/Matrix[1 0 0 1 " , format("%f %f", -dx,-dy), "]",
3146   "/Resources " , resources,
3147   "/Group " , format(pdfetcs.resfmt, on),
3148 } :gsub(decimals,rmzeros)
3149 on = update_pdfobjs(attr, streamtext)
3150 os = format("<</SMask<</S/Luminosity/G %s>>>>", format(pdfetcs.resfmt, on))
3151 end
3152 on, new = update_pdfobjs(os)
3153 local key = add_extgs_resources(on,new)
3154 start_pdf_code()
3155 pdf_literalcode("/%s gs", key)
3156 if fd_stop then return "standalone" end
3157 return "start"
3158 end
3159

```

Transparency Group

```

3160 pdfetcs.tr_group = { shifts = { } }
3161 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
3162 local function do_preobj_GRP (object, prescript)
3163   local grstate = prescript and prescript.gr_state
3164   if not grstate then return end
3165   local trgroup = pdfetcs.tr_group
3166   if grstate == "start" then
3167     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
3168     trgroup.isolated, trgroup.knockout, trgroup.off = false, false, false
3169     trgroup.wrapped = false
3170     for _,v in ipairs(prescript.gr_type:gsub("%s",""):explode",+) do
3171       trgroup[v] = true
3172     end
3173     trgroup.bbox = prescript.mplibgroupbbox:explode": "
3174     put2output[["\beginpgroup\setbox\mplibscratchbox\hbox\bgroup\luamplibtagasgroupset]]
3175   elseif grstate == "stop" then
3176     local llx,lly,urx,ury = tableunpack(trgroup.bbox)
3177     put2output(tableconcat{

```

```

3178     "\egroup",
3179     format("\wd\mplibscratchbox %fbp", urx-llx),
3180     format("\ht\mplibscratchbox %fbp", ury-lly),
3181     "\dp\mplibscratchbox 0pt",
3182   })
3183   local grattr
3184   if trgroup.off then
3185     grattr = ""
3186   else
3187     local on = update_pdfobjs(format("<</S/Transparency/I %s/K %s>>",
3188                                   trgroup.isolated, trgroup.knockout))
3189     grattr = format("/Group %s", pdfetcs.resfmt:format(on))
3190   end
3191   local res = gather_resources("")
3192   local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
3193   if pdfmode then
3194     if trgroup.wrapped then
3195       put2output(tableconcat{
3196         "\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3197         "/BBox[" , bbox, "]" } resources{", res, "}"\mplibscratchbox",
3198         "\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}",
3199       })
3200     end
3201     put2output(tableconcat{
3202       "\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
3203       "/BBox[" , bbox, "]", grattr, "]" } resources{", res, "}"\mplibscratchbox",
3204       "\luamplibtagasgroupput{" , trgroup.name, "}" ,
3205       [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
3206       [[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]],
3207       [[\box\mplibscratchbox]],
3208       "}"\endgroup",
3209       "\expandafter\def\csname luamplib.group.", trgroup.name, "\endcsname{" ,
3210       "\setbox\mplibscratchbox\hbox{\hskip",-llx,"bp\raise",-lly,"bp\hbox{" ,
3211       "\useboxresource \the\lastsavedboxresourceindex",
3212       "}"\wd\mplibscratchbox",urx-llx,"bp\ht\mplibscratchbox",ury-lly,"bp",
3213       "\box\mplibscratchbox}",
3214     })
3215   else
3216     if trgroup.wrapped then
3217       trgroup.cnt = (trgroup.cnt or 0) + 1
3218       local objname = format("@mplibrgr%s", trgroup.cnt)
3219       put2output(tableconcat{
3220         "\special{pdf:boxobj " , objname, " bbox " , bbox, "}",
3221         "\unhbox\mplibscratchbox",
3222         "\special{pdf:put @resources <<" , res, ">>}",
3223         "\special{pdf:exobj}",
3224         "\setbox\mplibscratchbox\hbox{\special{pdf:uxobj " , objname, "}"},
3225       })
3226     end

```

```

3227   trgroup.cnt = (trgroup.cnt or 0) + 1
3228   local objname = format("@mplibrtrgr%s", trgroup.cnt)
3229   put2output(tableconcat{
3230     "\\special{pdf:bxobj ", objname, " bbox ", bbox, "}",
3231     "\\unhbox\\mplibrscratchbox",
3232     "\\special{pdf:put @resources <<", res, ">>}",
3233     "\\special{pdf:exobj <<", grattr, ">>}",
3234     "\\luamplibtagasgroupput{"..trgroup.name,"}{"..",
3235     "\\special{pdf:uxobj ", objname, "}",
3236     "}"..endgroup",
3237   })
3238   token.set_macro("luamplib.group."..trgroup.name, tableconcat{
3239     "\\setbox\\mplibrscratchbox\\hbox{\\hskip",-llx,"bp\\raise",-lly,"bp\\hbox{"..",
3240     "\\special{pdf:uxobj ", objname, "}",
3241     "}"..wd\\mplibrscratchbox",urx-llx,"bp\\ht\\mplibrscratchbox",ury-lly,"bp",
3242     "\\box\\mplibrscratchbox",
3243     }, "global")
3244   end
3245   trgroup.shifts[trgroup.name] = { llx, lly }
3246 end
3247 return grstate
3248 end
3249 function luamplib.registergroup (boxid, name, opts)
3250   if opts.asgroup and opts.asgroup:find"wrapped" then
3251     luamplib.registergroup(boxid, name, {bbox=opts.bbox, resources=opts.resources})
3252     run_tex_code{"\\setbox", boxid, "\\hbox bdir0{\\csname luamplib.group.", name, "\\endcsname}"}
3253     opts.asgroup = opts.asgroup:gsub("wrapped", "")
3254   end
3255   local box = texgetbox(boxid)
3256   local wd, ht, dp = node.getwhd(box)
3257   local is_mask = opts.asgroup and opts.asgroup:find"masking"
3258   local res = opts.resources or ""
3259   res = gather_resources(res)
3260   local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
3261   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
3262   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
3263   if opts.matrix and opts.matrix:find"%a" then
3264     local t = get_mp_matrix(opts.matrix)
3265     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3266   end
3267   local grtype = 3
3268   if opts.bbox then
3269     attr[#attr+1] = format("/BBox[%s]", opts.bbox)
3270     grtype = 2
3271   end
3272   local mpllx, mplly = get_macro'MPlLx', get_macro'MPlLy'
3273   if is_mask then
3274     local t = opts.matrix and opts.matrix:explode() or {1, 0, 0, 1, 0, 0}
3275     t[5], t[6] = t[5]+mpllx, t[6]+mplly

```

```

3276     opts.matrix = format("%f %f %f %f %f %f",tableunpack(t))
3277     mpllx, mplly = 0, 0
3278 end
3279 if opts.matrix then
3280     attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
3281     grtype = opts.bbox and 4 or 1
3282 end
3283 if opts.asgroup and not opts.asgroup:find"off" then
3284     local t = { isolated = false, knockout = false, masking = false }
3285     for _,v in ipairs(opts.asgroup:gsub("%s",""):explode",") do t[v] = true end
3286     local on
3287     if t.masking then
3288         on = update_pdfobjs(format("<</S/Transparency/CS%s>>", opts.colorspace or "/DeviceGray"))
3289     else
3290         local cs = opts.colorspace and ("/CS%s"):format(opts.colorspace) or ""
3291         on = update_pdfobjs(format("<</S/Transparency%s/I %s/K %s>>", cs, t.isolated, t.knockout))
3292     end
3293     attr[#attr+1] = format("/Group %s", pdfetcs.resfmt:format(on))
3294 end
3295 local trgroup = pdfetcs.tr_group
3296 trgroup.shifts[name] = { mpllx, mplly }
3297 local whd
3298 if pdfmode then
3299     attr = tableconcat(attr) :gsub(decimals,rmzeros)
3300     local index = tex.saveboxresource(boxid, attr, res, true, grtype)
3301     token.set_macro("luamplib.group.".name, tableconcat{
3302         "\\useboxresource ", index,
3303     }, "global")
3304     whd = format("%.3f %.3f 0", wd/factor, (ht+dp)/factor) :gsub(decimals,rmzeros)
3305 else
3306     trgroup.cnt = (trgroup.cnt or 0) + 1
3307     local objname = format("@mplibrtrgr%s", trgroup.cnt)
3308     texsprint {
3309         "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3310         "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3311         "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
3312         "\\special{pdf:bcontent}",
3313         "\\special{pdf:bobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
3314         "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
3315         "\\special{pdf:put @resources <<", res, ">>}",
3316         "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
3317         "\\special{pdf:econtent}}",
3318     }
3319     token.set_macro("luamplib.group.".name, tableconcat{
3320         "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uobj ", objname, "}}",
3321         "\\wd\\mplibscratchbox ", wd, "sp",
3322         "\\ht\\mplibscratchbox ", ht, "sp",
3323         "\\dp\\mplibscratchbox ", dp, "sp",
3324         "\\box\\mplibscratchbox",

```

```

3325   }, "global")
3326   whd = format("%.3f %.3f %.3f", wd/factor, ht/factor, dp/factor) :gsub(decimals,rmzeros)
3327   end
3328   info("w/h/d of group '%s': %s", name, whd)
3329 end
3330

```

luamplib.convert: flushing figures

```

3331 do
3332 local function stop_special_effects(fade,opaq,over)
3333   if fade then -- fading
3334     stop_pdf_code()
3335   end
3336   if opaq then -- opacity
3337     pdf_literalcode(opaq)
3338   end
3339   if over then -- color
3340     if over:find"pdf:bc" then
3341       put2output"\special{pdf:ec}"
3342     else
3343       put2output"\special{color pop}"
3344     end
3345   end
3346 end
3347

```

For parsing prescript materials.

```

3348 local function script2table(s)
3349   local t = {}
3350   for _,i in ipairs(s:explode("\13+")) do
3351     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
3352     if k and v and k ~= "" and not t[k] then
3353       t[k] = v
3354     end
3355   end
3356   return t
3357 end
3358

```

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

```

3359 local function pdf_textfigure(font,size,text,width,height,depth)
3360   text = text:gsub(".",function(c)
3361     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
3362   end)
3363   put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}", font, size, text, 0, 0)
3364 end
3365
3366 local bend_tolerance = 131/65536
3367

```

```

3368 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
3369
3370 local function pen_characteristics(object)
3371     local t = mplib.pen_info(object)
3372     rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
3373     divider = sx*sy - rx*ry
3374     return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
3375 end
3376
3377 local function concat(px, py) -- no tx, ty here
3378     return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
3379 end
3380
3381 local function curved(ith,pth)
3382     local d = pth.left_x - ith.right_x
3383     if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and
3384         abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
3385         d = pth.left_y - ith.right_y
3386         if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and
3387             abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
3388             return false
3389         end
3390     end
3391     return true
3392 end
3393
3394 local function flushnormalpath(path,open)
3395     local pth, ith
3396     for i=1,#path do
3397         pth = path[i]
3398         if not ith then
3399             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
3400             elseif curved(ith,pth) then
3401                 pdf_literalcode("%f %f %f %f %f c",
3402                     ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
3403             else
3404                 pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
3405             end
3406             ith = pth
3407         end
3408     if not open then
3409         local one = path[1]
3410         if curved(pth,one) then
3411             pdf_literalcode("%f %f %f %f %f %f c",
3412                 pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord )
3413             else
3414                 pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3415             end
3416         elseif #path == 1 then -- special case .. draw point

```

```

3417     local one = path[1]
3418     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
3419 end
3420 end
3421
3422 local function flushconcatpath(path,open)
3423     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
3424     local pth, ith
3425     for i=1,#path do
3426         pth = path[i]
3427         if not ith then
3428             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
3429         elseif curved(ith,pth) then
3430             local a, b = concat(ith.right_x,ith.right_y)
3431             local c, d = concat(pth.left_x,pth.left_y)
3432             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
3433         else
3434             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
3435         end
3436         ith = pth
3437     end
3438     if not open then
3439         local one = path[1]
3440         if curved(pth,one) then
3441             local a, b = concat(pth.right_x,pth.right_y)
3442             local c, d = concat(one.left_x,one.left_y)
3443             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
3444         else
3445             pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3446         end
3447     elseif #path == 1 then -- special case .. draw point
3448         local one = path[1]
3449         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
3450     end
3451 end
3452

```

Finally, flush figures by inserting PDF literals.

```

3453 local function flush (result,flusher)
3454     if result then
3455         local figures = result.fig
3456         if figures then
3457             for f=1, #figures do
3458                 info("flushing figure %s",f)
3459                 local figure = figures[f]
3460                 local objects = figure:objects()
3461                 local fignum = tonumber(figure:filename():match("[%d]+$") or figure:charcode() or 0)
3462                 local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3463                 local bbox = figure:boundingbox()

```

```

3464         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
3465         if urx < llx then

```

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

```

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

3466         else

```

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

```

3467         if tex_code_pre_mplib[f] then
3468             put2output(tex_code_pre_mplib[f])
3469         end
3470         pdf_startfigure(fignum,llx,lly,urx,ury)
3471         start_pdf_code()
3472         if objects then
3473             local savedpath = nil
3474             local savedhtap = nil
3475             for o=1,#objects do
3476                 local object      = objects[o]
3477                 local objecttype  = object.type

```

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

```

3478             local prescript      = object.prescript
3479             prescript = prescript and script2table(prescript) -- prescript is now a table
3480             local cr_over = do_preobj_CR(object,prescript) -- color
3481             local tr_opaq = do_preobj_TR(object,prescript) -- opacity
3482             local fading_ = do_preobj_FADE(object,prescript) -- fading
3483             local pattern_ = do_preobj_PAT(object,prescript) -- tiling pattern
3484             local shading_ = do_preobj_shading(object,prescript) -- shading pattern
3485             local trgroup = do_preobj_GRP(object,prescript) -- transparency group
3486             if prescript and prescript.mplibtexboxid then
3487                 put_tex_boxes(object,prescript)
3488             elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
3489             elseif objecttype == "start_clip" then
3490                 local evenodd = not object.istext and object.postscript == "evenodd"
3491                 start_pdf_code()
3492                 flushnormalpath(object.path,false)
3493                 pdf_literalcode(evenodd and "W* n" or "W n")
3494             elseif objecttype == "stop_clip" then
3495                 stop_pdf_code()
3496                 miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3497             elseif objecttype == "special" then

```

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

```

3498             if prescript and prescript.postmplibverbtex then
3499                 figcontents.post[#figcontents.post+1] = prescript.postmplibverbtex
3500             end

```

```

3501     elseif objecttype == "text" then
3502         local ot = object.transform -- 3,4,5,6,1,2
3503         start_pdf_code()
3504         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
3505         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
3506         stop_pdf_code()
3507     elseif not trgroup and fading_ ~= "stop" then
3508         local evenodd, collect, both = false, false, false
3509         local postscript = object.postscript
3510         if not object.istext then
3511             if postscript == "evenodd" then
3512                 evenodd = true
3513             elseif postscript == "collect" then
3514                 collect = true
3515             elseif postscript == "both" then
3516                 both = true
3517             elseif postscript == "eoboth" then
3518                 evenodd = true
3519                 both = true
3520             end
3521         end
3522         if collect then
3523             if not savedpath then
3524                 savedpath = { object.path or false }
3525                 savedhtap = { object.htap or false }
3526             else
3527                 savedpath[#savedpath+1] = object.path or false
3528                 savedhtap[#savedhtap+1] = object.htap or false
3529             end
3530         else

```

Removed from ConTeXt general: color stuff.

```

3531         local ml = object.miterlimit
3532         if ml and ml ~= miterlimit then
3533             miterlimit = ml
3534             pdf_literalcode("%f M",ml)
3535         end
3536         local lj = object.linejoin
3537         if lj and lj ~= linejoin then
3538             linejoin = lj
3539             pdf_literalcode("%i j",lj)
3540         end
3541         local lc = object.linecap
3542         if lc and lc ~= linecap then
3543             linecap = lc
3544             pdf_literalcode("%i J",lc)
3545         end
3546         local dl = object.dash
3547         if dl then

```

```

3548         local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
3549         if d ~= dashed then
3550             dashed = d
3551             pdf_literalcode(dashed)
3552         end
3553     elseif dashed then
3554         pdf_literalcode("[[] 0 d")
3555         dashed = false
3556     end
3557     local path = object.path
3558     local transformed, penwidth = false, 1
3559     local open = path and path[1].left_type and path[#path].right_type
3560     local pen = object.pen
3561     if pen then
3562         if pen.type == 'elliptical' then
3563             transformed, penwidth = pen_characteristics(object) -- boolean, value
3564             pdf_literalcode("%f w",penwidth)
3565             if objecttype == 'fill' then
3566                 objecttype = 'both'
3567             end
3568         else -- calculated by mplib itself
3569             objecttype = 'fill'
3570         end
3571     end
end

```

Added : shading

```

3572     local shade_no, shade_stroke, shade_cm = do_preobj_SH(object,prescript) -- shading
3573     if shade_no then
3574         pdf_literalcode"q /Pattern cs"
3575         objecttype = false
3576     end
3577     if transformed then
3578         start_pdf_code()
3579     end
3580     if path then
3581         if savedpath then
3582             for i=1,#savedpath do
3583                 local path = savedpath[i]
3584                 if transformed then
3585                     flushconcatpath(path,open)
3586                 else
3587                     flushnormalpath(path,open)
3588                 end
3589             end
3590             savedpath = nil
3591         end
3592         if transformed then
3593             flushconcatpath(path,open)
3594         else

```

```

3595         flushnormalpath(path,open)
3596     end
3597     if objecttype == "fill" then
3598         pdf_literalcode(evenodd and "h f*" or "h f")
3599     elseif objecttype == "outline" then
3600         if both then
3601             pdf_literalcode(evenodd and "h B*" or "h B")
3602         else
3603             pdf_literalcode(open and "S" or "h S")
3604         end
3605     elseif objecttype == "both" then
3606         pdf_literalcode(evenodd and "h B*" or "h B")
3607     end
3608 end
3609 if transformed then
3610     stop_pdf_code()
3611 end
3612 local path = object.htap

```

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

```

3613     if path then
3614         if transformed then
3615             start_pdf_code()
3616         end
3617         if savedhtap then
3618             for i=1,#savedhtap do
3619                 local path = savedhtap[i]
3620                 if transformed then
3621                     flushconcatpath(path,open)
3622                 else
3623                     flushnormalpath(path,open)
3624                 end
3625             end
3626             savedhtap = nil
3627             evenodd = true
3628         end
3629         if transformed then
3630             flushconcatpath(path,open)
3631         else
3632             flushnormalpath(path,open)
3633         end
3634         if objecttype == "fill" then
3635             pdf_literalcode(evenodd and "h f*" or "h f")
3636         elseif objecttype == "outline" then
3637             pdf_literalcode(open and "S" or "h S")
3638         elseif objecttype == "both" then
3639             pdf_literalcode(evenodd and "h B*" or "h B")
3640         end
3641         if transformed then

```

```

3642         stop_pdf_code()
3643     end
3644 end

```

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

```

3645     if shade_no then -- shading
3646         pdf_literalcode("W%s %s %s/MPLibSh%s sh Q",
3647             evenodd and "*" or "",
3648             shade_stroke and "s" or "n",
3649             shade_cm and shade_cm.." cm " or "",
3650             shade_no)
3651     end
3652 end
3653 end
3654 if fading_ == "start" then
3655     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}
3656 elseif trgroup == "start" then
3657     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
3658 elseif fading_ == "stop" then
3659     local se = pdfetcs.fading.specialeffects
3660     if se then stop_special_effects(se[1], se[2], se[3]) end
3661 elseif trgroup == "stop" then
3662     local se = pdfetcs.tr_group.specialeffects
3663     if se then stop_special_effects(se[1], se[2], se[3]) end
3664 else
3665     stop_special_effects(fading_, tr_opaq, cr_over)
3666 end
3667 if fading_ or trgroup then -- extgs resetted
3668     miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
3669 end
3670 end
3671 end
3672 stop_pdf_code()
3673 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimex code.

```

3674     for _,v in ipairs(figcontents) do
3675         if type(v) == "table" then
3676             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
3677         else
3678             texsprint(v)
3679         end
3680     end
3681     if #figcontents.post > 0 then texsprint(figcontents.post) end
3682     figcontents = { post = { } }
3683 end
3684 end
3685 end
3686 end
3687 end

```

```

3688
3689 function luamplib.convert (result, flusher)
3690   flush(result, flusher)
3691   return true -- done
3692 end
3693 end
3694
3695 function luamplib.colorconverter (cr)
3696   local n = #cr
3697   if n == 4 then
3698     local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
3699     return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
3700   elseif n == 3 then
3701     local r, g, b = cr[1], cr[2], cr[3]
3702     return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
3703   else
3704     local s = cr[1]
3705     return format("%.3f g %.3f G",s,s), "0 g 0 G"
3706   end
3707 end

```

2.2 \TeX package

First we need to load some packages.

```

3708 \ifcsname ProvidesPackage\endcsname

```

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

```

3709 \NeedsTeXFormat{LaTeX2e}
3710 \ProvidesPackage{luamplib}
3711   [2026/07/09 v2.42.3 mplib package for LuaTeX]
3712 \fi
3713 \ifdefined\newluafunction\else
3714   \input ltluatex
3715 \fi

```

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

```

3716 \ifnum\outputmode=0
3717   \ifdefined\AddToHookNext
3718     \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
3719     \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
3720     \def\luamplibateveryshipout{\AddToHook{shipout/background}}
3721   \else
3722     \input atbegshi.sty
3723     \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
3724     \let\luamplibatfirstshipout\AtBeginShipoutFirst
3725     \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}

```

```
3726 \fi
3727 \fi
```

Loading of lua code.

```
3728 \directlua{require("luamplib")}
```

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

```
3729 \ifx\pdfoutput\undefined
```

```
3730 \let\pdfoutput\outputmode
```

```
3731 \fi
```

```
3732 \ifx\pdfliteral\undefined
```

```
3733 \protected\def\pdfliteral{\pdfextension literal}
```

```
3734 \fi
```

Set the format for METAPOST.

```
3735 \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.

```
3736 \ifnum\pdfoutput>0
```

```
3737 \let\mplibtoPDF\pdfliteral
```

```
3738 \else
```

```
3739 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
```

```
3740 \ifcsname PackageInfo\endcsname
```

```
3741 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
```

```
3742 \else
```

```
3743 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
```

```
3744 \fi
```

```
3745 \fi
```

To make mplibcode typeset always in horizontal mode.

```
3746 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
```

```
3747 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
```

```
3748 \mplibnoforcehmode
```

Catcode. We want to allow comment sign in mplibcode.

```
3749 \def\mplibsetupcatcodes{%
```

```
3750 %catcode`\{=12 %catcode`\}=12
```

```
3751 \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
```

```
3752 \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^M=12
```

```
3753 }
```

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

```
3754 \ifnum\outputmode>0 %
```

```
3755 \def\mplibputtextbox#1#2#3#4{%
```

```
3756 \pdfextension save\relax
```

```
3757 \vbox to 0pt{\vss
```

```
3758 \hbox bdir0 to 0pt{\kern#2bp\pdfextension setmatrix{#4}\raise\dp#1\copy#1\hss}%
```

```
3759 \kern#3bp}%
```

```
3760 \pdfextension restore\relax}
```

```
3761 \else
```

```

3762 \def\mplibputtextbox#1#2#3#4{%
3763   \special{pdf:btrans matrix #4 #2 #3}%
3764   \vbox to 0pt{\vss\hbox bdir0 to 0pt{\raise\dp#1\copy#1\hss}}%
3765   \special{pdf:etrans}}
3766 \fi

      use Transparency Group

3767 \protected\def\usemplibgroup#1#\{\usemplibgroupmain}
3768 \def\usemplibgroupmain#1{%
3769   \prependtomplibbox\hbox dir TL\bgroup
3770   \csname luamplib.group.#1\endcsname
3771   \egroup
3772 }
3773 \protected\def\mplibgroup#1{%
3774   \begingroup
3775   \def\MPllx{0}\def\MPlly{0}%
3776   \def\mplibgroupname{#1}%
3777   \mplibgroupgetnexttok
3778 }
3779 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
3780 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok= }
3781 \def\mplibgroupbranch{%
3782   \ifx [\nexttok
3783     \expandafter\mplibgroupopts
3784   \else
3785     \ifx\mplibsptoken\nexttok
3786       \expandafter\expandafter\expandafter\mplibgroupskipspace
3787     \else
3788       \let\mplibgroupoptions\empty
3789       \expandafter\expandafter\expandafter\mplibgroupmain
3790     \fi
3791   \fi
3792 }
3793 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
3794 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
3795 \protected\def\endmplibgroup{\egroup
3796   \directlua{ luamplib.registergroup(
3797     \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
3798   )}%
3799   \endgroup
3800 }

      Patterns

3801 {\def\:\global\let\mplibsptoken= } \: }
3802 \protected\def\mplipattern#1{%
3803   \begingroup
3804   \def\MPllx{0}\def\MPlly{0}%
3805   \def\mplipatternname{#1}%
3806   \mplipatterngetnexttok
3807 }

```

```

3808 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
3809 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok=}
3810 \def\mplibpatternbranch{%
3811   \ifx [\nexttok
3812     \expandafter\mplibpatternopts
3813   \else
3814     \ifx\mplibsptoken\nexttok
3815       \expandafter\expandafter\expandafter\mplibpatternskipsspace
3816     \else
3817       \let\mplibpatternoptions\empty
3818       \expandafter\expandafter\expandafter\mplibpatternmain
3819     \fi
3820   \fi
3821 }
3822 \def\mplibpatternopts[#1]{%
3823   \def\mplibpatternoptions{#1}%
3824   \mplibpatternmain
3825 }
3826 \def\mplibpatternmain{%
3827   \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
3828 }
3829 \protected\def\endmppattern{%
3830   \egroup
3831   \directlua{ luamplib.registerpattern(
3832     \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
3833   )}%
3834   \endgroup
3835 }

```

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

```

3836 \def\mpfiginstancename{@mpfig}
3837 \protected\def\mpfig{%
3838   \begingroup
3839   \futurelet\nexttok\mplibmpfigbranch
3840 }
3841 \def\mplibmpfigbranch{%
3842   \ifx *\nexttok
3843     \expandafter\mplibprempfig
3844   \else
3845     \ifx [\nexttok
3846       \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
3847     \else
3848       \expandafter\expandafter\expandafter\mplibmainmpfig
3849     \fi
3850   \fi
3851 }
3852 \def\mplibgobbleoptsmfig[#1]{\mplibmainmpfig}
3853 \def\mplibmainmpfig{%
3854   \begingroup

```

```

3855 \mplibsetupcatcodes
3856 \mplibdomainmpfig
3857 }
3858 \long\def\mplibdomainmpfig#1\endmpfig{%
3859 \endgroup
3860 \directlua{
3861   local legacy = luamplib.legacyverbatim
3862   local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
3863   local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
3864   luamplib.legacyverbatim = false
3865   luamplib.everymplib["\mpfiginstancename"] = ""
3866   luamplib.everyendmplib["\mpfiginstancename"] = ""
3867   luamplib.process_mplibcode(
3868     "beginfig(0) "..everympfig.." "..[====[\unexpanded{#1}]====].." "..everyendmpfig.." endfig;",
3869     "\mpfiginstancename")
3870   luamplib.legacyverbatim = legacy
3871   luamplib.everymplib["\mpfiginstancename"] = everympfig
3872   luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3873 }%
3874 \endgroup
3875 }
3876 \def\mplibprempfig#1{%
3877 \begingroup
3878 \mplibsetupcatcodes
3879 \mplibdoprempfig
3880 }
3881 \long\def\mplibdoprempfig#1\endmpfig{%
3882 \endgroup
3883 \directlua{
3884   local legacy = luamplib.legacyverbatim
3885   local everympfig = luamplib.everymplib["\mpfiginstancename"]
3886   local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
3887   luamplib.legacyverbatim = false
3888   luamplib.everymplib["\mpfiginstancename"] = ""
3889   luamplib.everyendmplib["\mpfiginstancename"] = ""
3890   luamplib.process_mplibcode([====[\unexpanded{#1}]====], "\mpfiginstancename")
3891   luamplib.legacyverbatim = legacy
3892   luamplib.everymplib["\mpfiginstancename"] = everympfig
3893   luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
3894 }%
3895 \endgroup
3896 }
3897 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

3898 \unless\ifcsname ver@luamplib.sty\endcsname
3899 \def\mplibcodegetinstancename[#1]{\xdef\currentmpinstancename{#1}\mplibcodeindeed}
3900 \protected\def\mplibcode{%
3901 \begingroup

```

```

3902   \futurelet\nexttok\mplibcodebranch
3903 }
3904 \def\mplibcodebranch{%
3905   \ifx [\nexttok
3906     \expandafter\mplibcodegetinstancename
3907   \else
3908     \global\let\currentmpinstancename\empty
3909     \expandafter\mplibcodeindeed
3910   \fi
3911 }
3912 \def\mplibcodeindeed{%
3913   \begingroup
3914   \mplibsetupcatcodes
3915   \mplibdocode
3916 }
3917 \long\def\mplibdocode#1\endmplibcode{%
3918   \endgroup
3919   \directlua[luamplib.process_mplibcode([===[\unexpanded{#1}]===],"\currentmpinstancename")]%
3920   \endgroup
3921 }
3922 \protected\def\endmplibcode{endmplibcode}
3923 \else

```

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

```

3924 \newenvironment{mplibcode}[1][{}%
3925   \xdef\currentmpinstancename{#1}%
3926   \mplibtmptoks{\ltxdomplibcode
3927 }{}
3928 \def\ltxdomplibcode{%
3929   \begingroup
3930   \mplibsetupcatcodes
3931   \ltxdomplibcodeindeed
3932 }
3933 \def\mplib@mplibcode{mplibcode}
3934 \long\def\ltxdomplibcodeindeed#1\end#2{%
3935   \endgroup
3936   \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
3937   \def\mplibtemp@a{#2}%
3938   \ifx\mplib@mplibcode\mplibtemp@a
3939     \directlua[luamplib.process_mplibcode([===[\the\mplibtmptoks]===],"\currentmpinstancename")]%
3940     \end{mplibcode}%
3941   \else
3942     \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
3943     \expandafter\ltxdomplibcode
3944   \fi
3945 }
3946 \fi

```

User settings.

```

3947 \def\mplibshowlog#1{\directlua{

```

```

3948   local s = string.lower("#1")
3949   if s == "enable" or s == "true" or s == "yes" then
3950     luamplib.showlog = true
3951   else
3952     luamplib.showlog = false
3953   end
3954 }}
3955 \def\mpliblegacybehavior#1{\directlua{
3956   local s = string.lower("#1")
3957   if s == "enable" or s == "true" or s == "yes" then
3958     luamplib.legacyverbatim = true
3959   else
3960     luamplib.legacyverbatim = false
3961   end
3962 }}
3963 \def\mplibverbatim#1{\directlua{
3964   local s = string.lower("#1")
3965   if s == "enable" or s == "true" or s == "yes" then
3966     luamplib.verbatiminput = true
3967   else
3968     luamplib.verbatiminput = false
3969   end
3970 }}
3971 \newtoks\mplibmptoks

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

3972 \ifcsname ver@luamplib.sty\endcsname
3973   \protected\def\everymplib{%
3974     \begingroup
3975     \mplibsetupcatcodes
3976     \mplibdoeverymplib
3977   }
3978   \protected\def\everyendmplib{%
3979     \begingroup
3980     \mplibsetupcatcodes
3981     \mplibdoeveryendmplib
3982   }
3983   \newcommand\mplibdoeverymplib[2][]{%
3984     \endgroup
3985     \directlua{
3986       luamplib.everymplib["#1"] = [===[\unexpanded{#2}]===]
3987     }%
3988   }
3989   \newcommand\mplibdoeveryendmplib[2][]{%
3990     \endgroup
3991     \directlua{
3992       luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===]
3993     }%
3994   }

```

```

3995 \else
3996 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
3997 \protected\def\everymplib#1#1{%
3998 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
3999 \begingroup
4000 \mplibsetupcatcodes
4001 \mplibdoeverymplib
4002 }
4003 \long\def\mplibdoeverymplib#1{%
4004 \endgroup
4005 \directlua{
4006 luamplib.everymplib["\currentmpinstancename"] = [====[\unexpanded{#1}]====]
4007 }%
4008 }
4009 \protected\def\everyendmplib#1#1{%
4010 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
4011 \begingroup
4012 \mplibsetupcatcodes
4013 \mplibdoeveryendmplib
4014 }
4015 \long\def\mplibdoeveryendmplib#1{%
4016 \endgroup
4017 \directlua{
4018 luamplib.everyendmplib["\currentmpinstancename"] = [====[\unexpanded{#1}]====]
4019 }%
4020 }
4021 \fi

```

TeX macros for dimen/color

```

4022 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
4023 \def\mpcolor#1#1{\domplibcolor{#1}}
4024 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

4025 \def\mplibnumbersystem#1{\directlua{
4026 local t = "#1"
4027 if t == "binary" then t = "decimal" end
4028 luamplib.numbersystem = t
4029 }}

```

Settings for .mp cache files.

```

4030 \def\mplibmakenocache#1{\mplibdomakenocache #1,\stop,}
4031 \def\mplibdomakenocache#1,{%
4032 \ifx\empty#1\empty
4033 \expandafter\mplibdomakenocache
4034 \else
4035 \ifx\stop#1\else
4036 \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
4037 \expandafter\expandafter\expandafter\mplibdomakenocache
4038 \fi

```

```

4039 \fi
4040 }
4041 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,\stop,}
4042 \def\mplibdocancelnocache#1,{%
4043 \ifx\empty#1\empty
4044 \expandafter\mplibdocancelnocache
4045 \else
4046 \ifx\stop#1\else
4047 \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
4048 \expandafter\expandafter\expandafter\mplibdocancelnocache
4049 \fi
4050 \fi
4051 }
4052 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

4053 \def\mplibtexttextlabel#1{\directlua{
4054 local s = string.lower("#1")
4055 if s == "enable" or s == "true" or s == "yes" then
4056 luamplib.texttextlabel = true
4057 else
4058 luamplib.texttextlabel = false
4059 end
4060 }}
4061 \def\mplibcodeinherit#1{\directlua{
4062 local s = string.lower("#1")
4063 if s == "enable" or s == "true" or s == "yes" then
4064 luamplib.codeinherit = true
4065 else
4066 luamplib.codeinherit = false
4067 end
4068 }}
4069 \def\mplibglobaltexttext#1{\directlua{
4070 local s = string.lower("#1")
4071 if s == "enable" or s == "true" or s == "yes" then
4072 luamplib.globaltexttext = true
4073 else
4074 luamplib.globaltexttext = false
4075 end
4076 }}

```

The followings are from ConT_EXt general, mostly.

We use a dedicated scratchbox.

```

4077 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

4078 \def\mplibstarttoPDF#1#2#3#4{%
4079 \prependtomplibbox
4080 \hbox dir TLT\bgroup
4081 \xdef\MP1lx{#1}\xdef\MP1ly{#2}%

```

```

4082 \xdef\MPurx{#3}\xdef\MPury{#4}%
4083 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4084 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4085 \parskip0pt%
4086 \leftskip0pt%
4087 \parindent0pt%
4088 \everypar{}%
4089 \setbox\mplibscratchbox\ vbox\ bgroup
4090 \noindent
4091 }
4092 \def\mplibstoptoPDF{%
4093 \par
4094 \egroup %
4095 \setbox\mplibscratchbox\ hbox %
4096 {\hskip-\MPllx bp%
4097 \raise-\MPlly bp%
4098 \box\mplibscratchbox}%
4099 \setbox\mplibscratchbox\ vbox to \MPheight
4100 {\vfill
4101 \hsize\MPwidth
4102 \wd\mplibscratchbox0pt%
4103 \ht\mplibscratchbox0pt%
4104 \dp\mplibscratchbox0pt%
4105 \box\mplibscratchbox}%
4106 \wd\mplibscratchbox\MPwidth
4107 \ht\mplibscratchbox\MPheight
4108 \box\mplibscratchbox
4109 \egroup
4110 }

```

Text items have a special handler.

```

4111 \def\mplibtexttext#1#2#3#4#5{%
4112 \begingroup
4113 \setbox\mplibscratchbox\ hbox
4114 {\font\temp=#1 at #2bp%
4115 \temp
4116 #3}%
4117 \setbox\mplibscratchbox\ hbox
4118 {\hskip#4 bp%
4119 \raise#5 bp%
4120 \box\mplibscratchbox}%
4121 \wd\mplibscratchbox0pt%
4122 \ht\mplibscratchbox0pt%
4123 \dp\mplibscratchbox0pt%
4124 \box\mplibscratchbox
4125 \endgroup
4126 }

```

Input luamplib.cfg when it exists.

```

4127 \openin0=luamplib.cfg

```

```

4128 \ifeof0 \else
4129 \closein0
4130 \input luamplib.cfg
4131 \fi

```

Code for tagpdf

```

4132 \def\luamplibtagtextboxset#1#2{#2}
4133 \let\luamplibnotagtextboxset\luamplibtagtextboxset
4134 \let\luamplibtagasgroupset\relax
4135 \let\luamplibtagasgroupput\luamplibtagtextboxset
4136 \ifcsname SuspendTagging\endcsname\else\endinput\fi
4137 \ifcsname ver@tagpdf.sty\endcsname \else
4138 \ExplSyntaxOn
4139 \keys_define:nn{luamplib/tagging}
4140 {
4141 ,alt .code:n = { }
4142 ,actualtext .code:n = { }
4143 ,artifact .code:n = { }
4144 ,text .code:n = { }
4145 ,off .code:n = { }
4146 ,tag .code:n = { }
4147 ,adjust-BBox .code:n = { }
4148 ,tagging-setup .code:n = { }
4149 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4150 ,instancename .meta:n = { instance = {#1} }
4151 ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4152 }
4153 \RenewDocumentCommand\mplibcode{0{}}
4154 {
4155 \tl_gclear:N \currentmpinstancename
4156 \keys_set:ne{luamplib/tagging}{#1}
4157 \mplibtmptoks{}\ltxdomplibcode
4158 }
4159 \cs_set_eq:NN \mplibaltext \use_none:n
4160 \cs_set_eq:NN \mplibactualtext \use_none:n

```

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

```

4161 \tl_put_left:Nn \mplibstoptoPDF \@newlistfalse
4162 \ExplSyntaxOff
4163 \endinput\fi
4164 \ExplSyntaxOn
4165 \tl_new:N \l__luamplib_tag_envname_tl
4166 \tl_new:N \l__luamplib_tag_alt_tl
4167 \tl_new:N \l__luamplib_tag_alt_dflt_tl
4168 \tl_new:N \l__luamplib_tag_actual_tl

```

```

4169 \tl_new:N \l__luamplib_tag_struct_tl
4170 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
4171 \bool_new:N \l__luamplib_tag_usetext_bool
4172 \bool_new:N \l__luamplib_tag_bboxcorr_bool
4173 \seq_new:N \l__luamplib_tag_bboxcorr_seq
4174 \tl_new:N \l__luamplib_tag_bbox_draw_tl
4175 \tl_new:N \l__luamplib_BBox_llx_tl
4176 \tl_new:N \l__luamplib_BBox_lly_tl
4177 \tl_new:N \l__luamplib_BBox_urx_tl
4178 \tl_new:N \l__luamplib_BBox_ury_tl
4179 \msg_new:nnn {luamplib}{figure-text-reuse}
4180 {
4181   tex-text~box~#1~probably~is~incorrectly~tagged.~
4182   Reusing~a~box~in~text~mode~is~strongly~discouraged.~
4183   Check~the~resulting~PDF.
4184 }
4185 \msg_new:nnn {luamplib}{mplibgroup-text-mode}
4186 {
4187   mplibgroup~'#1'~probably~is~incorrectly~tagged.~
4188   Using~mplibgroup~with~text~mode~is~not~recommended.~
4189   Check~the~resulting~PDF.
4190 }
4191 \msg_new:nnn{luamplib}{alt-text-missing}
4192 {
4193   Alternate~text~for~#1~is~missing.~
4194   Using~the~default~value~'#2'~instead.
4195 }

```

Sockets for tex-text boxes.

```

4196 \socket_new:nn{tagsupport/luamplib/texttext/set}{2}
4197 \socket_new:nn{tagsupport/luamplib/texttext/put}{2}
4198 \socket_new_plug:nnn{tagsupport/luamplib/texttext/set}{default}
4199 {

```

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.

```

4200 \bool_if:NTF \l__luamplib_tag_usetext_bool
4201 {
4202   \tag_mc_end_push:
4203   \tag_struct_begin:n{tag=NonStruct, stash, parent-tag=text}
4204   \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.

```

4205   \tag_mc_begin:n{tag=text}
4206   #2
4207   \tag_mc_end:
4208   \tag_struct_end:
4209   \tag_mc_begin_pop:n{}
4210 }
4211 {

```

```

4212 \tag_suspend:n{\luamplibtagtextboxset}
4213 #2
4214 \tag_resume:n{\luamplibtagtextboxset}
4215 }
4216 }
4217 \socket_new_plug:nnn{tagsupport/luamplib/texttext/put}{default}
4218 {
4219 \bool_lazy_and:nnTF
4220 { \l__luamplib_tag_usertext_bool }
4221 { \cs_if_free_p:c {luamplib.taggedbox.#1} }
4222 {
4223 \tag_resume:n{\mplibputtextbox}
4224 \tag_mc_end:
4225 \cs_if_exist:cTF {luamplib.taggedbox.#1}
4226 {
4227 \exp_args:Nc \tag_struct_use_num:n {luamplib.taggedbox.#1}
4228 #2
4229 \cs_undefine:c {luamplib.taggedbox.#1}
4230 }
4231 {
4232 \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
4233 \tag_mc_begin:n{ }
4234 \int_set:Nn \l_tmpa_int {#1}
4235 \tag_mc_reset_box:N \l_tmpa_int
4236 #2
4237 \tag_mc_end:
4238 }
4239 \tag_mc_begin:n{artifact}
4240 }
4241 {
4242 \int_set:Nn \l_tmpa_int {#1}
4243 \tag_mc_reset_box:N \l_tmpa_int
4244 #2
4245 }
4246 }
4247 \socket_assign_plug:nn{tagsupport/luamplib/texttext/set}{default}
4248 \socket_assign_plug:nn{tagsupport/luamplib/texttext/put}{default}
4249 \cs_set_nopar:Npn \luamplibtagtextboxset
4250 {
4251 \tag_socket_use:nnn{luamplib/texttext/set}
4252 }

```

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

```

4253 \cs_set_nopar:Npn \luamplibnotagtextboxset #1 #2
4254 {
4255 \bool_set_eq:NN \l_tmpa_bool \l__luamplib_tag_usertext_bool
4256 \bool_set_false:N \l__luamplib_tag_usertext_bool
4257 \tag_socket_use:nnn{luamplib/texttext/set}{#1}{#2}

```

```

4258 \cs_gset_nopar:cpn {luamplib.notaggedbox.#1}{#1}
4259 \bool_set_eq:NN \l__luamplib_tag_usetext_bool \l_tmpa_bool
4260 }
4261 \sys_if_output_pdf:TF
4262 {
4263 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4264 {
4265 \pdfextension save\relax
4266 \vbox to 0pt{\vss
4267 \hbox bdir0 to 0pt{\kern #2bp \pdfextension setmatrix {#4}
4268 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}
4269 \kern #3bp}
4270 \pdfextension restore\relax
4271 }
4272 }
4273 {
4274 \cs_set_nopar:Npn \mplibputtextbox #1 #2 #3 #4
4275 {
4276 \special{pdf:btrans~matrix~#4~#2~#3}
4277 \vbox to 0pt{\vss\hbox bdir0 to 0pt{
4278 \socket_use:nnn{tagsupport/luamplib/texttext/put}{#1}{\raise\dp#1\copy#1}\hss}}
4279 \special{pdf:etrans}
4280 }
4281 }

```

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

```

4282 \cs_set_nopar:Npn \luamplibtagasgroupset
4283 {
4284 \bool_set_false:N \l__luamplib_tag_usetext_bool
4285 }
4286 \cs_set_nopar:Npn \luamplibtagasgroupput
4287 {
4288 \bool_if:NT \l__luamplib_tag_usetext_bool { \tag_resume:n{\luamplibtagasgroupput} }
4289 \tag_socket_use:nnn{luamplib/mplibgroup/put}
4290 }

```

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

```

4291 \socket_new:nn{tagsupport/luamplib/mplibgroup/put}{2}
4292 \socket_new_plug:nnn{tagsupport/luamplib/mplibgroup/put}{default}
4293 {
4294 \cs_if_free:cT {luamplib.mplibgroup.text.#1}
4295 {
4296 \msg_warning:nnn {luamplib} {mplibgroup-text-mode} {#1}
4297 \cs_gset_nopar:cpn {luamplib.mplibgroup.text.#1} {#1}
4298 }
4299 \tag_mc_end:
4300 \tag_mc_begin:n{tag=text}
4301 #2
4302 \tag_mc_end:

```

```

4303 \tag_mc_begin:n{artifact}
4304 }
4305 \socket_assign_plug:nn{tagsupport/luamplib/mplibgroup/put}{default}

```

A macro for BBox attribute

```

4306 \cs_set_nopar:Npn \__luamplib_tag_bbox_attribute:n #1
4307 {
4308   \tl_set:Nc \l_tmpa_tl {luamplib.BBox.\tag_get:n{struct_num}}
4309   \tex_savepos:D
4310   \property_record:ee{\l_tmpa_tl}{xpos,ypos}
4311   \tl_set:Nc \l__luamplib_BBox_llx_tl
4312     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{xpos}{0}sp } }
4313   \tl_set:Nc \l__luamplib_BBox_lly_tl
4314     { \dim_to_decimal_in_bp:n { \property_ref:een {\l_tmpa_tl}{ypos}{0}sp - \dp#1 } }
4315   \tl_set:Nc \l__luamplib_BBox_urx_tl
4316     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_llx_tl bp + \wd#1 } }
4317   \tl_set:Nc \l__luamplib_BBox_ury_tl
4318     { \dim_to_decimal_in_bp:n { \l__luamplib_BBox_lly_tl bp + \ht#1 + \dp#1 } }
4319   \bool_if:NT \l__luamplib_tag_bboxcorr_bool
4320   {
4321     \int_zero:N \l_tmpa_int
4322     \tl_map_inline:nn
4323     {
4324       \l__luamplib_BBox_llx_tl
4325       \l__luamplib_BBox_lly_tl
4326       \l__luamplib_BBox_urx_tl
4327       \l__luamplib_BBox_ury_tl
4328     }
4329     {
4330       \int_incr:N \l_tmpa_int
4331       \tl_set:Nc ##1
4332       {
4333         \fp_eval:n
4334         {
4335           ##1
4336           +
4337           \dim_to_decimal_in_bp:n { \seq_item:NV \l__luamplib_tag_bboxcorr_seq \l_tmpa_int }
4338         }
4339       }
4340     }
4341   }
4342   \tag_struct_gput:ene {\tag_get:n{struct_num}} {attribute}
4343   {
4344     /O /Layout /BBox [
4345       \l__luamplib_BBox_llx_tl\c_space_tl
4346       \l__luamplib_BBox_lly_tl\c_space_tl
4347       \l__luamplib_BBox_urx_tl\c_space_tl
4348       \l__luamplib_BBox_ury_tl
4349     ]

```

```

4350 }
4351 \bool_if:NT \l__tag_graphic_debug_bool
4352 {
4353   \iow_log:e
4354   {
4355     luamplib/tagging~debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
4356     \l__luamplib_BBox_llx_tl\c_space_tl
4357     \l__luamplib_BBox_lly_tl\c_space_tl
4358     \l__luamplib_BBox_urx_tl\c_space_tl
4359     \l__luamplib_BBox_ury_tl
4360   }
4361   \sys_if_output_pdf:TF
4362   {
4363     \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
4364     {
4365       \pdfextension save\relax
4366       \opacity_select:n{0.5} \color_select:n{red}
4367       \pdfextension literal~text
4368       {
4369         \l__luamplib_BBox_llx_tl\c_space_tl
4370         \l__luamplib_BBox_lly_tl\c_space_tl
4371         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4372         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4373         re~f
4374       }
4375       \pdfextension restore\relax
4376     }
4377   }
4378   {
4379     \tl_set:Ne \l__luamplib_tag_bbox_draw_tl
4380     {
4381       \special{pdf:bcontent}
4382       \opacity_select:n{0.5} \color_select:n{red}
4383       \special{pdf:code~
4384         1~0~0~1~
4385         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{xpos}{0}sp + \wd#1 }~
4386         -\dim_to_decimal_in_bp:n { \property_ref:een{\l_tmpa_tl}{ypos}{0}sp }~
4387         cm
4388       }
4389       \special{pdf:code~
4390         \l__luamplib_BBox_llx_tl\c_space_tl
4391         \l__luamplib_BBox_lly_tl\c_space_tl
4392         \fp_eval:n { \l__luamplib_BBox_urx_tl - \l__luamplib_BBox_llx_tl }~
4393         \fp_eval:n { \l__luamplib_BBox_ury_tl - \l__luamplib_BBox_lly_tl }~
4394         re~f
4395       }
4396       \special{pdf:econtent}
4397     }
4398   }

```

```
4399 }
4400 }
```

Sockets for main process

```
4401 \socket_new:nn{tagsupport/luamplib/figure/begin}{1}
4402 \socket_new:nn{tagsupport/luamplib/figure/end}{2}
4403 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{transparent}{#2}
4404 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{alt}
4405 {
4406   \tag_mc_end_push:
4407   \tl_if_empty:NT\l__luamplib_tag_alt_tl
4408   {
4409     \tl_if_empty:eTF{#1}
4410     { \tl_set:Nn \l__luamplib_tag_alt_tl {metapost~figure} }
4411     { \tl_set:Ne \l__luamplib_tag_alt_tl {metapost~figure~\text_purify:n{#1}} }
4412     \msg_warning:nnVV{luamplib}{alt-text-missing}
4413     \l__luamplib_tag_envname_tl \l__luamplib_tag_alt_tl
4414   }
4415   \tag_struct_begin:n
4416   {
4417     tag=\l__luamplib_tag_struct_tl,
4418     alt=\l__luamplib_tag_alt_tl,
4419   }
4420   \tag_mc_begin:n{}
4421 }
4422 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{alt}
4423 {
4424   \l__luamplib_tag_bbox_attribute:n {#1}
4425   #2
4426   \tl_use:N \l__luamplib_tag_bbox_draw_tl
4427   \tag_mc_end:
4428   \tag_struct_end:
4429   \tag_mc_begin_pop:n{}
4430 }
4431 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{actualtext}
4432 {
4433   \tag_mc_end_push:
4434   \tag_struct_begin:n
4435   {
4436     tag=Span,
4437     actualtext=\l__luamplib_tag_actual_tl,
4438   }
4439   \tag_mc_begin:n{}
4440 }
4441 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{actualtext}
4442 {
4443   #2
4444   \tag_mc_end:
4445   \tag_struct_end:
```

```

4446   \tag_mc_begin_pop:n{ }
4447 }
4448 \socket_new_plug:nnn{tagsupport/luamplib/figure/begin}{artifact}
4449 {
4450   \tag_mc_end_push:
4451   \tag_mc_begin:n{artifact}
4452 }
4453 \socket_new_plug:nnn{tagsupport/luamplib/figure/end}{artifact}
4454 {
4455   #2
4456   \tag_mc_end:
4457   \tag_mc_begin_pop:n{ }
4458 }

```

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

```

4459 \socket_new:nn{tagsupport/luamplib/figure/init}{0}
4460 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{alt}
4461 {
4462   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{alt}
4463   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{alt}
4464 }
4465 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{actualtext}
4466 {
4467   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{actualtext}
4468   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{actualtext}

```

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

```

4469   \prependtomplibbox \mplibnoforcehmode
4470   \mode_if_vertical:T { \noindent \aftergroup\par }
4471 }
4472 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{artifact}
4473 {
4474   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4475   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4476 }
4477 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{text}
4478 {
4479   \bool_set_true:N \l__luamplib_tag_usetext_bool
4480   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{artifact}
4481   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{artifact}
4482   \prependtomplibbox \mplibnoforcehmode
4483   \mode_if_vertical:T { \noindent \aftergroup\par }
4484 }
4485 \socket_new_plug:nnn{tagsupport/luamplib/figure/init}{off}
4486 {
4487   \socket_assign_plug:nn{tagsupport/luamplib/figure/begin}{noop}
4488   \socket_assign_plug:nn{tagsupport/luamplib/figure/end}{transparent}
4489 }
4490 \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}

```

Key-value options

```
4491 \keys_define:nn{luamplib/tagging}
4492 {
4493   ,alt .code:n =
4494   {
4495     \tl_set:Nn\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
4496     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4497   }
4498   ,actualtext .code:n =
4499   {
4500     \tl_set:Nn\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
4501     \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{actualtext}
4502   }
4503   ,artifact .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{artifact} }
4504   ,text .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{text} }
4505   ,off .code:n = { \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{off} }
4506   ,tag .code:n =
4507   {
4508     \str_case:nnF {#1}
4509     {
4510       {false} { \keys_set:nn {luamplib/tagging} {off} }
4511       {artifact} { \keys_set:nn {luamplib/tagging} {artifact} }
4512     }
4513     {
4514       \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
4515       \socket_assign_plug:nn{tagsupport/luamplib/figure/init}{alt}
4516     }
4517   }
4518   ,adjust-BBox .code:n =
4519   {
4520     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
4521     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt~0pt}
4522   }
4523   ,tagging-setup .code:n = { \keys_set_known:nn {luamplib/tagging} {#1} }
4524 }
4525 \keys_define:nn {luamplib/instance}
4526 {
4527   ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
4528   ,instancename .meta:n = { instance = {#1} }
4529   ,unknown .code:n = { \tl_gset:NV \currentmpinstancename \l_keys_key_str }
4530 }
```

Redefine our macros

```
4531 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
4532 {
4533   \prependtomplibbox
4534   \hbox dir~TLT\bgroup
4535     \tag_socket_use:nn{luamplib/figure/begin}\l__luamplib_tag_alt_dflt_tl
4536     \xdef\MPLlx{#1}\xdef\MPlly{#2}%
```

```

4537 \xdef\MPurx{#3}\xdef\MPury{#4}%
4538 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
4539 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
4540 \parskip0pt
4541 \leftskip0pt
4542 \parindent0pt
4543 \everypar{}%
4544 \setbox\mplibscratchbox\vbox\bgroup
4545 \tag_suspend:n{\mplibstarttoPDF}
4546 \noindent
4547 }
4548 \cs_set_nopar:Npn \mplibstoptoPDF
4549 {
4550 \par
4551 \egroup
4552 \setbox\mplibscratchbox\hbox
4553 {\hskip-\MPllx bp
4554 \raise-\MPlly bp
4555 \box\mplibscratchbox}%
4556 \setbox\mplibscratchbox\vbox to \MPheight
4557 {\vfill
4558 \hsize\MPwidth
4559 \wd\mplibscratchbox0pt
4560 \ht\mplibscratchbox0pt
4561 \dp\mplibscratchbox0pt
4562 \box\mplibscratchbox}%
4563 \wd\mplibscratchbox\MPwidth
4564 \ht\mplibscratchbox\MPheight
4565 \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\box\mplibscratchbox}
4566 \egroup
4567 }
4568 \RenewDocumentCommand\mplibcode{0{}}
4569 {
4570 \tl_set:Nn \l__luamplib_tag_envname_tl {mplibcode}
4571 \tl_gclear:N \currentmpinstancename
4572 \keys_set_known:neN {luamplib/tagging} {#1} \l_tmpa_tl
4573 \keys_set:nV {luamplib/instance} \l_tmpa_tl
4574 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \currentmpinstancename
4575 \tag_socket_use:n{luamplib/figure/init}
4576 \mplibtmptoks{\ltxdomplibcode
4577 }
4578 \RenewDocumentCommand\mpfig{s 0{}}
4579 {
4580 \begingroup
4581 \tl_set:Nn \l__luamplib_tag_envname_tl {mpfig}
4582 \keys_set_known:ne {luamplib/tagging} {#2}
4583 \tl_set_eq:NN \l__luamplib_tag_alt_dflt_tl \mpfiginstancename
4584 \tag_socket_use:n{luamplib/figure/init}
4585 \IfBooleanTF{#1} { \mplibprempfig * }

```

```

4586             { \mplibmainmpfig }
4587 }
4588 \RenewDocumentCommand\usemplibgroup{0}{ m}
4589 {
4590   \begingroup
4591   \tl_set:Nn \l__luamplib_tag_envname_tl {usemplibgroup}
4592   \keys_set_known:ne {luamplib/tagging} {#1}
4593   \tag_socket_use:n{luamplib/figure/init}
4594   \prependtomplibbox\hbox dir~TLT\bgroup
4595     \tag_socket_use:nn{luamplib/figure/begin}{#2}
4596     \setbox\mplibscratchbox\hbox\bgroup
4597       \bool_if:NF \l__luamplib_tag_usertext_bool { \tag_suspend:n{\usemplibgroup} }
4598       \tag_socket_use:nnn{luamplib/mplibgroup/put}{#2}{\csname luamplib.group.#2\endcsname}
4599       \egroup
4600       \tag_socket_use:nnn{luamplib/figure/end}{\mplibscratchbox}{\unhbox\mplibscratchbox}
4601       \egroup
4602   \endgroup
4603 }

```

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

```

4604 \cs_new_nopar:Npn \mplibalttext #1
4605 {
4606   \tl_set:Ne \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
4607 }
4608 \cs_new_nopar:Npn \mplibactualtext #1
4609 {
4610   \tl_set:Ne \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
4611 }
4612 \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 passes on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

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

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder stating it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- 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.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- 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.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both is 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 license cannot impose that choice.

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

9. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

10. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

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

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

NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR RE-DISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

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 anyone 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 66, 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 subpackage 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.