

The luamplib package

Hans Hagen, Taco Hoekwater, Elie Roux, Philipp Gesang and Kim Dohyun
Current Maintainer: Kim Dohyun
Support: <https://github.com/lualatex/luamplib>

2024/11/25 v2.35.1

Abstract

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

1 Documentation

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

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

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

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

- possibility to use `btex ... etex` to typeset TeX code. `textext()` is a more versatile macro equivalent to `TEX()` from `TEX.mp`. `TEX()` is also allowed and is a synonym of `textext()`. The argument of `mplib`'s primitive `maketext` will also be processed by the same routine.
- possibility to use `verbatimtex ... etex`, though its behavior cannot be the same as the stand-alone `mpost`. Of course you cannot include `\documentclass`, `\usepackage` etc. When these TeX commands are found in `verbatimtex ... etex`, the entire code will be ignored. The treatment of `verbatimtex` command has changed a lot since v2.20: see [below § 1.1](#).
- in the past, the package required PDF mode in order to have some output. Starting with version 2.7 it works in DVI mode as well, though DVIPDFMx is the only DVI tool currently supported.

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

1.1 T_EX

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

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

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

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

N.B. As *metafun* is such a complicated format, we cannot support all the functionalities producing special effects provided by *metafun*. At least, however, transparency (actually opacity), shading (gradient colors) and transparency group are fully supported, and `outlinetext` is supported by our own alternative `mpliboutlinetext` (see [below § 1.2](#)).

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

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.

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

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

where `<string>` should be "" (empty), "isolated", "knockout", or "isolated, knockout". Beware that currently many of the PDF rendering applications, except Adobe Acrobat Reader, cannot properly render the isolated or knockout effect. Transparency group is available with *plain* format as well, with extended functionality. See [below § 1.2](#).

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

\mplibshowlog{enable|disable} Default: disable. When `\mplibshowlog{enable}`¹ 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`.

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

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

N.B. `\endgraf` should be used instead of `\par` inside `verbatimtex ... etex`.

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

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

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

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

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

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

From v2.35, however, the redefinition of `infont` operator has been revised: when the character slot of the text argument is less than 32 (control characters), or is equal to

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

35 (#), 36 (\$), 37 (%), 38 (&), 92 (\), 94 (^), 95 (_), 123 (t), 125 (j), 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 T_EX.

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

Separate METAPOST instances luamplib v2.22 has added the support for several named METAPOST instances in L^AT_EX mplibcode environment. Plain T_EX users also can use this functionality. The syntax for L^AT_EX is:

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

The behavior is as follows.

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

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

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

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

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

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

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

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

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

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

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

and the starred version is roughly the same as follows:

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

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

```
\everymplib[@mpfig]{ drawoptions(withcolor .5[red,white]); }
\mpfig* input boxes \endmpfig
\mpfig
  circleit.a(btex Box 1 etex); drawboxed(a);
\endmpfig
```

The instance name (default: `@mpfig`) can be changed by redefining `\mpfiginstancename`, after which a new `mplib` instance will start and code inheritance too will begin anew. `\let\mpfiginstancename\empty` will prevent code inheritance if `\mplibcodeinherit{true}` is not declared.

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

- `\mplibmakenocache{<filename>[,<filename>,...]}`
- `\mplibcancelnocache{<filename>[,<filename>,...]}`

where `<filename>` is a filename excluding `.mp` extension. Note that `.mp` files under `$TEXMFMAIN/metapost/base` and `$TEXMFMAIN/metapost/context/base` are already registered by default.

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

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

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

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

Tagged PDF When `tagpdf` package is loaded and activated, `mplibcode` environment accepts additional options for tagged PDF. The code related to this functionality is currently in experimental stage, not guaranteeing backward compatibility. Like the \TeX 's `picture` environment, available optional keys are `tag`, `alt`, `actualtext`, `artifact`, `debug` and `correct-BBox` (`texdoc latex-lab-graphic`). Additionally, `luamplib` provides its own text key.

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

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

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

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

`actualtext=...` starts a `Span` tag implicitly and sets an actual text as given. Horizontal mode is forced by `\noindent` command. BBox info will not be added. This key is intended for figures which can be represented by a character or a small sequence of characters. You can give actual text within METAPOST code as well: `VerbatimTeX ("mplibactualtext{...}")`;

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

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

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

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

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

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

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

1.2 METAPOST

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

mplibtexcolor ..., mplibrbgtexcolor ... `mplibtexcolor`, which accepts a string argument, 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` operator. For instance:

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

But the result may vary in its color model (gray/rgb/cmyk) according to the given T_EX color. (Spot colors are forced to cmyk model, so this operator is not recommended for spot colors.) Therefore the example shown above would raise a METAPOST error: `cmykcolor col;` should have been declared. By contrast, `mplibrbgtexcolor <string>` always returns `rgb` model expressions.

mplibgraphicstext ... `mplibgraphicstext` is a METAPOST operator, the effect of which is similar to that of ConTeXt's `graphicstext` or our own `mpliboutlinetext` (see [below](#)). However the syntax is somewhat different.

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

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

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

mplibglyph ... of ... From v2.30, we provide a new METAPOST operator `mplibglyph`, which returns a METAPOST picture containing outline paths of a glyph in opentype, true-type or type1 fonts. When a type1 font is specified, METAPOST primitive `glyph` will be called.

```
mplibglyph 50 of \fontid\font          % slot 50 of current font
mplibglyph "Q" of "TU/TeXGyrePagella(0)/m/n/10" % font csname
mplibglyph "Q" of "texgyrepagella-regular.otf" % raw filename
mplibglyph "Q" of "Times.ttc(2)"          % subfont number
mplibglyph "Q" of "SourceHanSansK-VF.otf[Regular]" % instance name
```

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

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

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

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

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

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

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

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

An example:

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

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

Table 1: options for \mppattern

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

* in string type, numbers are separated by spaces

The available options are listed in Table 1.

For the sake of convenience, the width and height values of tiling patterns will be written down into the log file. (depth is always zero.) Users can refer to them for option setting.

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

When you use special effects such as transparency in a pattern, `resources` option is needed: for instance, `resources="/ExtGState 1 0 R"`. However, as `luamplib` automatically includes the resources of the current page, this option is not needed in most cases.

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

```

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

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

```

```

    endfor
  endfor
endfig;
\end{mplibcode}

```

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

```

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

```

... **withfademethod** ... This is a METAPOST operator which makes the color of an object gradiently transparent. The syntax is `<path>|<picture>withfademethod <string>`, the latter being either "linear" or "circular". Though it is similar to the `withshademethod` from *metafun*, the differences are: (1) the operand of `withfademethod` can be a picture as well as a path; (2) you cannot make gradient colors, but can only make gradient opacity.

Related macros to control optional values are:

`withfadeopacity` (*number, number*) sets the starting opacity and the ending opacity, default value being (1, 0). '1' denotes full color; '0' full transparency.

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

`withfadecenter` is a synonym of `withfadevector`.

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

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

An example:

```

\mpfig
picture mill;
mill = btex \includegraphics[width=100bp]{mill} etex;
draw mill

```

```

withfademethod "circular"
withfadecenter (center mill, center mill)
withfaderadius (20, 50)
withfadeopacity (1, 0)
;
\endmpfig

```

... **asgroup** ... As said [before](#), transparency group is available with *plain* as well as *metafun* format. The syntax is exactly the same: `<picture> | <path> asgroup "" | "isolated" | "knockout" | "isolated, knockout"`, which will return a METAPOST picture. It is called *Transparency Group* because the objects contained in the group are composited to produce a single object, so that outer transparency effect, if any, will be applied to the group as a whole, not to the individual objects cumulatively.

The additional feature provided by `luamplib` is that you can reuse the group as many times as you want in the \TeX code or in other METAPOST code chunks, with infinitesimal increase in the size of PDF file. For this functionality we provide \TeX and METAPOST macros as follows:

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

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

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

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

An example showing the difference between the \TeX and METAPOST commands:

```

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

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

```

Table 2: options for `\mplibgroup`

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

* in string type, numbers are separated by spaces

```

\mpfig
  usemplibgroup "mygroup" rotated 15
    withprescript "tr_transparency=0.5";
  draw (left--right) scaled 10;
  draw (up--down) scaled 10;
\endmpfig

```

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

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

```

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

\usemplibgroup{mygrx}

\mpfig
  usemplibgroup "mygrx" scaled 1.5
    withprescript "tr_transparency=0.5" ;
\endmpfig

```

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

When `asgroup` option, including empty string, is not given, a normal form XObject will be generated rather than a transparency group. Thus the individual objects, not the XObject as a whole, will be affected by outer transparency command.

As shown in the example, you can reuse the `mplibgroup` once defined using the TeX command `\usemplibgroup` or the METAPOST command `usemplibgroup`. The behavior of these commands is the same as that described [above](#).

1.3 Lua

runscript ... Using the primitive `runscript <string>`, you can run a Lua code chunk from METAPOST side and get some METAPOST code returned by Lua if you want. As the functionality is provided by the `mplib` library itself, `luamplib` does not have much to say about it.

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

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

```
\begin{mplibcode}[instance1]
  boolean b; b = 1 > 2;
  numeric n; n = 3;
  string s; s = "MetaPost";
  path p; p = unitsquare;
\end{mplibcode}

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

Lua function `luamplib.process_mplibcode` Users can execute a METAPOST code chunk from Lua side by using this function:

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

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

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

2 Implementation

2.1 Lua module

```
1
2 luatexbase.provides_module {
```

Table 3: elements in luamplib table (partial)

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

```

3 name      = "luamplib",
4 version   = "2.35.1",
5 date      = "2024/11/25",
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. ConT_EXt uses `metapost`.

```

9 luamplib      = luamplib or { }
10 local luamplib = luamplib
11
12 local format, abs = string.format, math.abs
13
14 Use our own function for warn/info/err.
15 local function termorlog (target, text, kind)
16   if text then
17     local mod, write, append = "luamplib", texio.write_nl, texio.write
18     kind = kind
19     or kind == "term" and "Warning (more info in the log)"
20     or kind == "log" and "Info"
21     or kind == "term and log" and "Warning"
22     or "Error"
23     target = kind == "Error" and "term and log" or target
24     local t = text:explode"\n+"
25     write(target, format("Module %s %s:", mod, kind))
26     if #t == 1 then
27       append(target, format(" %s", t[1]))
28     else
29       for _,line in ipairs(t) do
30         write(target, line)
31       end
32       write(target, format("(%s) ", mod))
33     end
34     append(target, format(" on input line %s", tex.inputlineno))
35     write(target, "")

```

```

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

```

This module is a stripped down version of libraries that are used by Con \TeX T. Provide a few “shortcuts” expected by the code.

```

50 local tableconcat = table.concat
51 local tableinsert = table.insert
52 local tableunpack = table.unpack
53 local teksprint   = tex.sprint
54 local texgettoks  = tex.gettoks
55 local texgetbox   = tex.getbox
56 local texruntoks  = tex.runtoks
57 if not texruntoks then
58   err("Your LuaTeX version is too old. Please upgrade it to the latest")
59 end
60 local is_defined = token.is_defined
61 local get_macro  = token.get_macro
62 local mplib     = require('mplib')
63 local kpse      = require('kpse')
64 local lfs       = require('lfs')
65 local lfsattributes = lfs.attributes
66 local lfsisdir   = lfs.isdir
67 local lfsmkdir   = lfs.mkdir
68 local lfstouch   = lfs.touch
69 local 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.mkdirs or function(path)
86   local full = ""
87   for sub in path:gmatch("(/*[^\\"/]+)") do
88     full = full .. sub
89     lfsmkdir(full)
90   end
91 end
92

```

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

```

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

```

Some basic METAPOST files not necessary to make cache files.

```

131 local noneedtoreplace = {
132   ["boxes.mp"] = true, -- ["format.mp"] = true,
133   ["graph.mp"] = true, ["marith.mp"] = true, ["mfplain.mp"] = true,
134   ["mpost.mp"] = true, ["plain.mp"] = true, ["rboxes.mp"] = true,

```

```

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

    format.mp is much complicated, so specially treated.
148 local function replaceformatmp(file,newfile,ofmodify)
149   local fh = ioopen(file,"r")
150   if not fh then return file end
151   local data = fh:read("*all"); fh:close()
152   fh = ioopen(newfile,"w")
153   if not fh then return file end
154   fh:write(
155     "let normalinfont = infont;\n",
156     "primarydef str infont name = rawtexttext(str) enddef;\n",
157     data,
158     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
159     "vardef Fexp_(expr x) = rawtexttext(\"$\^{\"&decimal x&\"}$\") enddef;\n",
160     "let infont = normalinfont;\n"
161   ); fh:close()
162   lfstouch(newfile,currenttime,ofmodify)
163   return newfile
164 end

```

Replace btex ... etex and verbatimtex ... etex in input files, if needed.

```

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

```

“etex” must be preceded by a space and followed by a space or semicolon as specified in

Lua \TeX manual, which is not the case of standalone METAPOST though.

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

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

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

```

```

236 end
237 return file
238 end
239 end
240

```

Create and load `mplib` instances. We do not support ancient version of `mplib` any more. (Don't know which version of `mplib` started to support `make_text` and `run_script`; let the users find it.)

```

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

```

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

```

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

```

v2.9 has introduced the concept of "code inherit"

```

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

```

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

```

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

```

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

```

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

```

```

280     info(log)
281   end
282 end
283 return log
284 end
285 end

```

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

```

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

```

Make use of `make_text` and `run_script`, which will co-operate with Lua_{T_EX}'s `tex.runtoks` or other Lua functions. And we provide `numbersystem` option since v2.4. See <https://github.com/lualatex/luamplib/issues/21>.

```

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

```

Append our own METAPOST preamble to the preamble above.

```

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

```

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

```

313 local function process (data, instancename)
314   local currfmt
315   if instancename and instancename ~= "" then
316     currfmt = instancename
317     has_instancename = true
318   else
319     currfmt = tableconcat{
320       currentformat,
321       luamplib.numbersystem or "scaled",
322       tostring(luamplib.texttextlabel),
323       tostring(luamplib.legacyverbatimtex),
324   }

```

```

325   has_instancename = false
326 end
327 local mpx = mplibinstances[currfmt]
328 local standalone = not (has_instancename or luamplib.codeinherit)
329 if mpx and standalone then
330   mpx:finish()
331 end
332 local log = ""
333 if standalone or not mpx then
334   mpx, _, log = luamplibload(currentformat)
335   mplibinstances[currfmt] = mpx
336 end
337 local converted, result = false, {}
338 if mpx and data then
339   result = mpx:execute(data)
340   local log = reporterror(result, log)
341   if log then
342     if result.fig then
343       converted = luamplib.convert(result)
344     end
345   end
346 else
347   err"Mem file unloadable. Maybe generated with a different version of mplib?"
348 end
349 return converted, result
350 end
351

```

dvipdfmx is supported, though nobody seems to use it.

```

352 local pdfmode = tex.outputmode > 0
353

```

make_text and some run_script uses LuaTeX's tex.runtoks.

```

354 local catlatex = luatexbase.registernumber("catcodetable@latex")
355 local catat11 = luatexbase.registernumber("catcodetable@atletter")

```

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

```

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

```

Prepare 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.

```

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

```

For conversion of sp to bp.

```

360 local factor = 65536*(7227/7200)
361 local textext_fmt = 'image(addto currentpicture doublepath unitsquare \z
362   xscaled %f yscaled %f shifted (0,-%f) \z
363   withprescript "mplibtexboxid=%i:%f:%f")'
364 local function process_tex_text (str, maketext)
365   if str then

```

```

366 if not maketext then str = str:gsub("\\r.-$", "") end
367 local global = (has_instancename or luamplib.globaltext or luamplib.codeinherit)
368 and "\\global" or ""
369 local tex_box_id
370 if global == "" then
371     tex_box_id = texboxes.localid + 1
372     texboxes.localid = tex_box_id
373 else
374     local boxid = texboxes.globalid + 1
375     texboxes.globalid = boxid
376     run_tex_code(format("[\\expandafter\\newbox\\csname luamplib.box.%s\\endcsname]", boxid))
377     tex_box_id = tex.getcount'alloctionnumber'
378 end
379 run_tex_code(format("\\luamplibtagtextbegin{%i}%s\\setbox%i\\hbox{%s}\\luamplibtagtextend", tex_box_id, global,
380 local box = texgetbox(tex_box_id)
381 local wd = box.width / factor
382 local ht = box.height / factor
383 local dp = box.depth / factor
384 return textext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
385 end
386 return ""
387 end
388

```

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

```

389 local mplibcolorfmt = {
390     xcolor = tableconcat{
391         [[\\begingroup\\let\\XC@color\\relax]],
392         [[\\def\\set@color{\\global\\mplibmptoks\\expandafter{\\current@color}}]],
393         [[\\color%s\\endgroup]],
394     },
395     l3color = tableconcat{
396         [[\\begingroup\\def\\__color_select:N#1{\\expandafter\\__color_select:nn#1}]],
397         [[\\def\\__color_backend_select:nn#1#2{\\global\\mplibmptoks{#1 #2}}]],
398         [[\\def\\__kernel_backend_literal:e#1{\\global\\mplibmptoks\\expandafter{\\expanded{#1}}}],
399         [[\\color_select:n%s\\endgroup]],
400     },
401 }
402 local colfmt = is_defined'color_select:n' and "l3color" or "xcolor"
403 if colfmt == "l3color" then
404     run_tex_code{
405         "\\newcatcodetable\\luamplibcctabexplat",
406         "\\begingroup",
407         "\\catcode`@=11 ",
408         "\\catcode`_=11 ",
409         "\\catcode`:=11 ",
410         "\\savecatcodetable\\luamplibcctabexplat",
411         "\\endgroup",
412     }
413 end
414 local ccexplat = luatexbase.registernumber"luamplibcctabexplat"
415 local function process_color (str)
416     if str then
417         if not str:find("%b{") then

```

```

418     str = format("{%s}",str)
419 end
420 local myfmt = mplibcolorfmt[colfmt]
421 if colfmt == "l3color" and is_defined"color" then
422     if str:find("%b[]") then
423         myfmt = mplibcolorfmt.xcolor
424     else
425         for _,v in ipairs(str:match"{(.+)":explode"!") do
426             if not v:find("^%s*d+%s*$") then
427                 local pp = get_macro(format("l_color_named_%s_prop",v))
428                 if not pp or pp == "" then
429                     myfmt = mplibcolorfmt.xcolor
430                     break
431                 end
432             end
433         end
434     end
435 end
436 run_tex_code(myfmt:format(str), ccexplat or catat11)
437 local t = texgettoks"mplibtmptoks"
438 if not pdfmode and not t:find"^pdf" then
439     t = t:gsub("%a+ (.+)", "pdf:bc [%1]")
440 end
441 return format('1 withprescript "mpliboverridecolor=%s"', t)
442 end
443 return ""
444 end
445
446 for \mpdim or mplibdimen
447 local function process_dimen (str)
448     if str then
449         str = str:gsub("{(.+)","%1")
450         run_tex_code(format([[mplibtmptoks\expandafter{\the\dimexpr %s\relax}]], str))
451         return format("begingroup %s endgroup", texgettoks"mplibtmptoks")
452     end
453     return ""
454 end

```

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

```

455 local function process_verbatimex_text (str)
456     if str then
457         run_tex_code(str)
458     end
459     return ""
460 end
461

```

For legacy verbatimex process. verbatimex ... etex before beginfig() is not ignored, but the \TeX code is inserted just before the mplib box. And \TeX code inside beginfig() ... endfig is inserted after the mplib box.

```

462 local tex_code_pre_mplib = {}
463 luamplib.figid = 1
464 luamplib.in_the_fig = false

```



```

465 local function process_verbatimtex_prefig (str)
466   if str then
467     tex_code_pre_mplib[luamplib.figid] = str
468   end
469   return ""
470 end
471 local function process_verbatimtex_infig (str)
472   if str then
473     return format('special "postmplibverbtex=%s";', str)
474   end
475   return ""
476 end
477
478 local runscript_funcs = {
479   luamplibtext    = process_tex_text,
480   luamplibcolor  = process_color,
481   luamplibdimen  = process_dimen,
482   luamplibprefig = process_verbatimtex_prefig,
483   luamplibinfig  = process_verbatimtex_infig,
484   luamplibverbtex = process_verbatimtex_text,
485 }
486

```

For *metafun* format. see issue #79.

```

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

```

metafun 2021-03-09 changes crashes luamplib.

```

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

```

A function from ConTEXt general.

```

503 local function mpprint(buffer,...)
504   for i=1,select("#",...) do
505     local value = select(i,...)
506     if value ~= nil then
507       local t = type(value)
508       if t == "number" then
509         buffer[#buffer+1] = format("%.16f",value)
510       elseif t == "string" then
511         buffer[#buffer+1] = value
512       elseif t == "table" then
513         buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
514       else -- boolean or whatever

```

```

515     buffer[#buffer+1] = tostring(value)
516   end
517 end
518 end
519 end
520 function luamplib.runscript (code)
521   local id, str = code:match("(.-){(.*)}")
522   if id and str then
523     local f = runscript_funcs[id]
524     if f then
525       local t = f(str)
526       if t then return t end
527     end
528   end
529   local f = loadstring(code)
530   if type(f) == "function" then
531     local buffer = {}
532     function mp.print(...)
533       mpprint(buffer,...)
534     end
535     local res = {f()}
536     buffer = tableconcat(buffer)
537     if buffer and buffer ~= "" then
538       return buffer
539     end
540     buffer = {}
541     mpprint(buffer, tableunpack(res))
542     return tableconcat(buffer)
543   end
544   return ""
545 end
546
547   make_text must be one liner, so comment sign is not allowed.
548 local function protecttexcontents (str)
549   return str:gsub("\\%", "\\0PerCent\0")
550         :gsub("%%.\n", "")
551         :gsub("%%.-$", "")
552         :gsub("%zPerCentz", "\\%")
553         :gsub("\\r.-$", "")
554         :gsub("%s+", " ")
555 end
556 luamplib.legacyverbatimex = true
557 function luamplib.maketext (str, what)
558   if str and str ~= "" then
559     str = protecttexcontents(str)
560     if what == 1 then
561       if not str:find("\\documentclass"..name_e) and
562         not str:find("\\begin%s*(document}") and
563         not str:find("\\documentstyle"..name_e) and
564         not str:find("\\usepackage"..name_e) then
565         if luamplib.legacyverbatimex then
566           if luamplib.in_the_fig then
567             return process_verbatimex_infig(str)
568           else

```

```

568         return process_verbatimtex_prefig(str)
569     end
570     else
571         return process_verbatimtex_text(str)
572     end
573 end
574 else
575     return process_tex_text(str, true) -- bool is for 'char13'
576 end
577 end
578 return ""
579 end
580
    luamplib's METAPOST color operators
581 local function colorsplit (res)
582 local t, tt = { }, res:gsub("[%[%]]", "", 2):explode()
583 local be = tt[1]:find"^%d" and 1 or 2
584 for i=be, #tt do
585     if not tonumber(tt[i]) then break end
586     t[#t+1] = tt[i]
587 end
588 return t
589 end
590
591 luamplib.gettexcolor = function (str, rgb)
592 local res = process_color(str):match'"mpliboverridecolor=(.+)"'
593 if res:find" cs " or res:find"@pdf.obj" then
594     if not rgb then
595         warn("%s is a spot color. Forced to CMYK", str)
596     end
597     run_tex_code({
598         "\\color_export:nnN{",
599         str,
600         "}{" ,
601         rgb and "space-sep-rgb" or "space-sep-cmyk",
602         "}\mplib@tempa",
603     }, ccexplat)
604     return get_macro"mplib@tempa":explode()
605 end
606 local t = colorsplit(res)
607 if #t == 3 or not rgb then return t end
608 if #t == 4 then
609     return { 1 - math.min(1, t[1]+t[4]), 1 - math.min(1, t[2]+t[4]), 1 - math.min(1, t[3]+t[4]) }
610 end
611 return { t[1], t[1], t[1] }
612 end
613
614 luamplib.shadecolor = function (str)
615 local res = process_color(str):match'"mpliboverridecolor=(.+)"'
616 if res:find" cs " or res:find"@pdf.obj" then -- spot color shade: 13 only

```

An example of spot color shading:

```

\documentclass{article}
\usepackage{luamplib}

```

```

\mplibsetformat{metafun}
\ExplSyntaxOn
\color_model_new:nnn { pantone3005 }
{ Separation }
{ name = PANTONE~3005~U ,
  alternative-model = cmyk ,
  alternative-values = {1, 0.56, 0, 0}
}
\color_set:nnn{spotA}{pantone3005}{1}
\color_set:nnn{spotB}{pantone3005}{0.6}
\color_model_new:nnn { pantone1215 }
{ Separation }
{ name = PANTONE~1215~U ,
  alternative-model = cmyk ,
  alternative-values = {0, 0.15, 0.51, 0}
}
\color_set:nnn{spotC}{pantone1215}{1}
\color_model_new:nnn { pantone2040 }
{ Separation }
{ name = PANTONE~2040~U ,
  alternative-model = cmyk ,
  alternative-values = {0, 0.28, 0.21, 0.04}
}
\color_set:nnn{spotD}{pantone2040}{1}
\ExplSyntaxOff
\begin{document}
\begin{mplibcode}
beginfig(1)
  fill unitsquare xyscaled (\mpdim\textwidth,1cm)
    withshademethod "linear"
    withshadevector (0,1)
    withshadestep (
      withshadefraction .5
      withshadecolors ("spotB","spotC")
    )
    withshadestep (
      withshadefraction 1
      withshadecolors ("spotC","spotD")
    )
  ;
endfig;
\end{mplibcode}
\end{document}

```

another one: user-defined DeviceN colorspace

```

\DocumentMetadata{ }
\documentclass{article}
\usepackage{luamplib}
\mplibsetformat{metafun}
\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
  withshademethod "linear"
  withshadecolors ("purepantone","pureblack")
  ;
\endmpfig
\end{document}

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

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

    luamplib's mplibgraphicstext operator
635 local emboldenfonts = { }
636 local function getemboldenwidth (curr, fakebold)
637   local width = emboldenfonts.width
638   if not width then
639     local f
640     local function getglyph(n)
641       while n do
642         if n.head then
643           getglyph(n.head)
644         elseif n.font and n.font > 0 then
645           f = n.font; break
646         end
647         n = node.getnext(n)
648       end

```

```

649     end
650     getglyph(curr)
651     width = font.getcopy(f or font.current()).size * fakebold / factor * 10
652     emboldenfonts.width = width
653 end
654 return width
655 end
656 local function getrulewhatsit (line, wd, ht, dp)
657 line, wd, ht, dp = line/1000, wd/factor, ht/factor, dp/factor
658 local pl
659 local fmt = "%f w %f %f %f %f re %s"
660 if pdfmode then
661     pl = node.new("whatsit", "pdf_literal")
662     pl.mode = 0
663 else
664     fmt = "pdf:content ".fmt
665     pl = node.new("whatsit", "special")
666 end
667 pl.data = fmt:format(line, 0, -dp, wd, ht+dp, "B") :gsub(decimals, rmzeros)
668 local ss = node.new"glue"
669 node.setglue(ss, 0, 65536, 65536, 2, 2)
670 pl.next = ss
671 return pl
672 end
673 local function getrulemetric (box, curr, bp)
674 local running = -1073741824
675 local wd,ht,dp = curr.width, curr.height, curr.depth
676 wd = wd == running and box.width or wd
677 ht = ht == running and box.height or ht
678 dp = dp == running and box.depth or dp
679 if bp then
680     return wd/factor, ht/factor, dp/factor
681 end
682 return wd, ht, dp
683 end
684 local function embolden (box, curr, fakebold)
685 local head = curr
686 while curr do
687     if curr.head then
688         curr.head = embolden(curr, curr.head, fakebold)
689     elseif curr.replace then
690         curr.replace = embolden(box, curr.replace, fakebold)
691     elseif curr.leader then
692         if curr.leader.head then
693             curr.leader.head = embolden(curr.leader, curr.leader.head, fakebold)
694         elseif curr.leader.id == node.id"rule" then
695             local glue = node.effective_glue(curr, box)
696             local line = getemboldenwidth(curr, fakebold)
697             local wd,ht,dp = getrulemetric(box, curr.leader)
698             if box.id == node.id"hlist" then
699                 wd = glue
700             else
701                 ht, dp = 0, glue
702             end

```

```

703     local pl = getrulewhatsit(line, wd, ht, dp)
704     local pack = box.id == node.id"hlist" and node.hpack or node.vpack
705     local list = pack(pl, glue, "exactly")
706     head = node.insert_after(head, curr, list)
707     head, curr = node.remove(head, curr)
708     end
709 elseif curr.id == node.id"rule" and curr.subtype == 0 then
710     local line = getemboldenwidth(curr, fakebold)
711     local wd,ht,dp = getrulemetric(box, curr)
712     if box.id == node.id"vlist" then
713         ht, dp = 0, ht+dp
714     end
715     local pl = getrulewhatsit(line, wd, ht, dp)
716     local list
717     if box.id == node.id"hlist" then
718         list = node.hpack(pl, wd, "exactly")
719     else
720         list = node.vpack(pl, ht+dp, "exactly")
721     end
722     head = node.insert_after(head, curr, list)
723     head, curr = node.remove(head, curr)
724 elseif curr.id == node.id"glyph" and curr.font > 0 then
725     local f = curr.font
726     local key = format("%s:%s",f,fakebold)
727     local i = emboldenfonts[key]
728     if not i then
729         local ft = font.getfont(f) or font.getcopy(f)
730         if pdfmode then
731             width = ft.size * fakebold / factor * 10
732             emboldenfonts.width = width
733             ft.mode, ft.width = 2, width
734             i = font.define(ft)
735         else
736             if ft.format ~= "opentype" and ft.format ~= "truetype" then
737                 goto skip_type1
738             end
739             local name = ft.name:gsub("'",''):gsub('$','')
740             name = format('%s;embolden=%s;',name,fakebold)
741             _, i = fonts.constructors.readanddefine(name,ft.size)
742         end
743         emboldenfonts[key] = i
744     end
745     curr.font = i
746 end
747 ::skip_type1::
748 curr = node.getnext(curr)
749 end
750 return head
751 end
752 local function graphictextcolor (col, filldraw)
753 if col:find"^[%d%.:]+$" then
754     col = col:explode"."
755     for i=1,#col do
756         col[i] = format("%.3f", col[i])

```

```

757 end
758 if pdfmode then
759   local op = #col == 4 and "k" or #col == 3 and "rg" or "g"
760   col[#col+1] = filldraw == "fill" and op or op:upper()
761   return tableconcat(col, " ")
762 end
763 return format("[%s]", tableconcat(col, " "))
764 end
765 col = process_color(col):match"mpliboverridecolor=(.+)'"
766 if pdfmode then
767   local t, tt = col:explode(), { }
768   local b = filldraw == "fill" and 1 or #t/2+1
769   local e = b == 1 and #t/2 or #t
770   for i=b,e do
771     tt[#tt+1] = t[i]
772   end
773   return tableconcat(tt, " ")
774 end
775 return col:gsub("^.- ", "")
776 end
777 luamplib.graphicstext = function (text, fakebold, fc, dc)
778   local fmt = process_tex_text(text):sub(1,-2)
779   local id = tonumber(fmt:match"mplibtextboxid=(%d+):")
780   emboldenfonts.width = nil
781   local box = texgetbox(id)
782   box.head = embolden(box, box.head, fakebold)
783   local fill = graphicstextcolor(fc, "fill")
784   local draw = graphicstextcolor(dc, "draw")
785   local bc = pdfmode and "" or "pdf:bc "
786   return format('%s withprescript "mpliboverridecolor=%s%s %s"', fmt, bc, fill, draw)
787 end
788

```

luamplib's mplibglyph operator

```

789 local function mperr (str)
790   return format("hide(errmessage %q)", str)
791 end
792 local function getangle (a,b,c)
793   local r = math.deg(math.atan(c.y-b.y, c.x-b.x) - math.atan(b.y-a.y, b.x-a.x))
794   if r > 180 then
795     r = r - 360
796   elseif r < -180 then
797     r = r + 360
798   end
799   return r
800 end
801 local function turning (t)
802   local r, n = 0, #t
803   for i=1,2 do
804     tableinsert(t, t[i])
805   end
806   for i=1,n do
807     r = r + getangle(t[i], t[i+1], t[i+2])
808   end
809   return r/360

```



```

810 end
811 local function glyphimage(t, fmt)
812   local q,p,r = {},{}
813   for i,v in ipairs(t) do
814     local cmd = v[#v]
815     if cmd == "m" then
816       p = {format('%s,%s',v[1],v[2])}
817       r = {{x=v[1],y=v[2]}}
818     else
819       local nt = t[i+1]
820       local last = not nt or nt[#nt] == "m"
821       if cmd == "l" then
822         local pt = t[i-1]
823         local seco = pt[#pt] == "m"
824         if (last or seco) and r[1].x == v[1] and r[1].y == v[2] then
825           else
826             tableinsert(p, format('--(%s,%s)',v[1],v[2]))
827             tableinsert(r, {x=v[1],y=v[2]})
828           end
829         if last then
830           tableinsert(p, '--cycle')
831         end
832       elseif cmd == "c" then
833         tableinsert(p, format('..controls(%s,%s)and(%s,%s)',v[1],v[2],v[3],v[4]))
834         if last and r[1].x == v[5] and r[1].y == v[6] then
835           tableinsert(p, '..cycle')
836         else
837           tableinsert(p, format('..(%s,%s)',v[5],v[6]))
838         if last then
839           tableinsert(p, '--cycle')
840         end
841         tableinsert(r, {x=v[5],y=v[6]})
842       end
843     else
844       return mperr"unknown operator"
845     end
846     if last then
847       tableinsert(q[ turning(r) > 0 and 1 or 2 ], tableconcat(p))
848     end
849   end
850 end
851 r = { }
852 if fmt == "opentype" then
853   for _,v in ipairs(q[1]) do
854     tableinsert(r, format('addto currentpicture contour %s;',v))
855   end
856   for _,v in ipairs(q[2]) do
857     tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
858   end
859 else
860   for _,v in ipairs(q[2]) do
861     tableinsert(r, format('addto currentpicture contour %s;',v))
862   end
863   for _,v in ipairs(q[1]) do

```

```

864     tableinsert(r, format('addto currentpicture contour %s withcolor background;',v))
865   end
866 end
867 return format('image(%s)', tableconcat(r))
868 end
869 if not table.tofile then require"lualibs-lpeg"; require"lualibs-table"; end
870 function luamplib.glyph (f, c)
871   local filename, subfont, instance, kind, shapedata
872   local fid = tonumber(f) or font.id(f)
873   if fid > 0 then
874     local fontdata = font.getfont(fid) or font.getcopy(fid)
875     filename, subfont, kind = fontdata.filename, fontdata.subfont, fontdata.format
876     instance = fontdata.specification and fontdata.specification.instance
877     filename = filename and filename:gsub("^harfloaded:", "")
878   else
879     local name
880     f = f:match"^%s*(.)%s*$"
881     name, subfont, instance = f:match"(.+)%((%d+)%)%[(.-)]$"
882     if not name then
883       name, instance = f:match"(.+)%[(.-)]$" -- SourceHanSansK-VF.otf[Heavy]
884     end
885     if not name then
886       name, subfont = f:match"(.+)%((%d+)%)$" -- Times.ttc(2)
887     end
888     name = name or f
889     subfont = (subfont or 0)+1
890     instance = instance and instance:lower()
891     for _,ftype in ipairs{"opentype", "truetype"} do
892       filename = kpse.find_file(name, ftype.." fonts")
893       if filename then
894         kind = ftype; break
895       end
896     end
897   end
898   if kind ~= "opentype" and kind ~= "truetype" then
899     f = fid and fid > 0 and tex.fontname(fid) or f
900     if kpse.find_file(f, "tfm") then
901       return format("glyph %s of %q", tonumber(c) or format("%q",c), f)
902     else
903       return mperr"font not found"
904     end
905   end
906   local time = lfsattributes(filename,"modification")
907   local k = format("shapes_%s(%s)[%s]", filename, subfont or "", instance or "")
908   local h = format(string.rep('%02x', 256/8), string.byte(sha2.digest256(k), 1, -1))
909   local newname = format("%s/%s.lua", cachedir or outputdir, h)
910   local newtime = lfsattributes(newname,"modification") or 0
911   if time == newtime then
912     shapedata = require(newname)
913   end
914   if not shapedata then
915     shapedata = fonts and fonts.handlers.otf.readers.loadshapes(filename,subfont,instance)
916     if not shapedata then return mperr"loadshapes() failed. luaotfload not loaded?" end
917     table.tofile(newname, shapedata, "return")

```

```

918  lfstouch(newname, time, time)
919  end
920  local gid = tonumber(c)
921  if not gid then
922    local uni = utf8.codepoint(c)
923    for i,v in pairs(shapedata.glyphs) do
924      if c == v.name or uni == v.unicode then
925        gid = i; break
926      end
927    end
928  end
929  if not gid then return mperr"cannot get GID (glyph id)" end
930  local fac = 1000 / (shapedata.units or 1000)
931  local t = shapedata.glyphs[gid].segments
932  if not t then return "image()" end
933  for i,v in ipairs(t) do
934    if type(v) == "table" then
935      for ii,vv in ipairs(v) do
936        if type(vv) == "number" then
937          t[i][ii] = format("%.0f", vv * fac)
938        end
939      end
940    end
941  end
942  kind = shapedata.format or kind
943  return glyphimage(t, kind)
944 end
945
mpliboutlinetext : based on mkiv's font-mps.lua
946 local rulefmt = "mpliboutlinetic[%i]:=image(addto currentpicture contour \z
947 unitsquare shifted - center unitsquare;) xscaled %f yscaled %f shifted (%f,%f);"
948 local outline_horz, outline_vert
949 function outline_vert (res, box, curr, xshift, yshift)
950  local b2u = box.dir == "LTL"
951  local dy = (b2u and -box.depth or box.height)/factor
952  local ody = dy
953  while curr do
954    if curr.id == node.id"rule" then
955      local wd, ht, dp = getrulemetric(box, curr, true)
956      local hd = ht + dp
957      if hd ~= 0 then
958        dy = dy + (b2u and dp or -ht)
959        if wd ~= 0 and curr.subtype == 0 then
960          res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+(ht-dp)/2)
961        end
962        dy = dy + (b2u and ht or -dp)
963      end
964    elseif curr.id == node.id"glue" then
965      local vwidth = node.effective_glue(curr,box)/factor
966      if curr.leader then
967        local curr, kind = curr.leader, curr.subtype
968        if curr.id == node.id"rule" then
969          local wd = getrulemetric(box, curr, true)
970          if wd ~= 0 then

```

```

971     local hd = vwidth
972     local dy = dy + (b2u and 0 or -hd)
973     if hd ~= 0 and curr.subtype == 0 then
974         res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+wd/2, yshift+dy+hd/2)
975     end
976 end
977 elseif curr.head then
978     local hd = (curr.height + curr.depth)/factor
979     if hd <= vwidth then
980         local dy, n, iy = dy, 0, 0
981         if kind == 100 or kind == 103 then -- todo: gleaders
982             local ady = abs(ody - dy)
983             local ndy = math.ceil(ady / hd) * hd
984             local diff = ndy - ady
985             n = (vwidth-diff) // hd
986             dy = dy + (b2u and diff or -diff)
987         else
988             n = vwidth // hd
989             if kind == 101 then
990                 local side = vwidth % hd / 2
991                 dy = dy + (b2u and side or -side)
992             elseif kind == 102 then
993                 iy = vwidth % hd / (n+1)
994                 dy = dy + (b2u and iy or -iy)
995             end
996         end
997         dy = dy + (b2u and curr.depth or -curr.height)/factor
998         hd = b2u and hd or -hd
999         iy = b2u and iy or -iy
1000         local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1001         for i=1,n do
1002             res = func(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1003             dy = dy + hd + iy
1004         end
1005     end
1006 end
1007 end
1008 dy = dy + (b2u and vwidth or -vwidth)
1009 elseif curr.id == node.id"kern" then
1010     dy = dy + curr.kern/factor * (b2u and 1 or -1)
1011 elseif curr.id == node.id"vlist" then
1012     dy = dy + (b2u and curr.depth or -curr.height)/factor
1013     res = outline_vert(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1014     dy = dy + (b2u and curr.height or -curr.depth)/factor
1015 elseif curr.id == node.id"hlist" then
1016     dy = dy + (b2u and curr.depth or -curr.height)/factor
1017     res = outline_horz(res, curr, curr.head, xshift+curr.shift/factor, yshift+dy)
1018     dy = dy + (b2u and curr.height or -curr.depth)/factor
1019 end
1020 curr = node.getnext(curr)
1021 end
1022 return res
1023 end
1024 function outline_horz (res, box, curr, xshift, yshift, discwd)

```

```

1025 local r2l = box.dir == "TRT"
1026 local dx = r2l and (discwd or box.width/factor) or 0
1027 local dirs = { { dir = r2l, dx = dx } }
1028 while curr do
1029   if curr.id == node.id"dir" then
1030     local sign, dir = curr.dir:match"(.)..."
1031     local level, newdir = curr.level, r2l
1032     if sign == "+" then
1033       newdir = dir == "TRT"
1034       if r2l ~= newdir then
1035         local n = node.getnext(curr)
1036         while n do
1037           if n.id == node.id"dir" and n.level+1 == level then break end
1038           n = node.getnext(n)
1039         end
1040         n = n or node.tail(curr)
1041         dx = dx + node.rangedimensions(box, curr, n)/factor * (newdir and 1 or -1)
1042       end
1043       dirs[level] = { dir = r2l, dx = dx }
1044     else
1045       local level = level + 1
1046       newdir = dirs[level].dir
1047       if r2l ~= newdir then
1048         dx = dirs[level].dx
1049       end
1050     end
1051     r2l = newdir
1052   elseif curr.char and curr.font and curr.font > 0 then
1053     local ft = font.getfont(curr.font) or font.getcopy(curr.font)
1054     local gid = ft.characters[curr.char].index or curr.char
1055     local scale = ft.size / factor / 1000
1056     local slant = (ft.slant or 0)/1000
1057     local extend = (ft.extend or 1000)/1000
1058     local squeeze = (ft.squeeze or 1000)/1000
1059     local expand = 1 + (curr.expansion_factor or 0)/1000000
1060     local xscale = scale * extend * expand
1061     local yscale = scale * squeeze
1062     dx = dx - (r2l and curr.width/factor*expand or 0)
1063     local xpos = dx + xshift + (curr.xoffset or 0)/factor
1064     local ypos = yshift + (curr.yoffset or 0)/factor
1065     local vertical = ft.shared and ft.shared.features.vertical and "rotated 90" or ""
1066     if vertical ~= "" then -- luatexko
1067       for _,v in ipairs(ft.characters[curr.char].commands or { }) do
1068         if v[1] == "down" then
1069           ypos = ypos - v[2] / factor
1070         elseif v[1] == "right" then
1071           xpos = xpos + v[2] / factor
1072         else
1073           break
1074         end
1075       end
1076     end
1077     local image
1078     if ft.format == "opentype" or ft.format == "truetype" then

```

```

1079     image = luamplib.glyph(curr.font, gid)
1080 else
1081     local name, scale = ft.name, 1
1082     local vf = font.read_vf(name, ft.size)
1083     if vf and vf.characters[gid] then
1084         local cmds = vf.characters[gid].commands or {}
1085         for _,v in ipairs(cmds) do
1086             if v[1] == "char" then
1087                 gid = v[2]
1088             elseif v[1] == "font" and vf.fonts[v[2]] then
1089                 name = vf.fonts[v[2]].name
1090                 scale = vf.fonts[v[2]].size / ft.size
1091             end
1092         end
1093     end
1094     image = format("glyph %s of %q scaled %f", gid, name, scale)
1095 end
1096 res[#res+1] = format("mpliboutlinepic[%i]:= %s xscaled %f yscaled %f slanted %f %s shifted (%f,%f);",
1097     #res+1, image, xscale, yscale, slant, vertical, xpos, ypos)
1098 dx = dx + (r2l and 0 or curr.width/factor*expand)
1099 elseif curr.replace then
1100     local width = node.dimensions(curr.replace)/factor
1101     dx = dx - (r2l and width or 0)
1102     res = outline_horz(res, box, curr.replace, xshift+dx, yshift, width)
1103     dx = dx + (r2l and 0 or width)
1104 elseif curr.id == node.id"rule" then
1105     local wd, ht, dp = getrulemetric(box, curr, true)
1106     if wd ~= 0 then
1107         local hd = ht + dp
1108         dx = dx - (r2l and wd or 0)
1109         if hd ~= 0 and curr.subtype == 0 then
1110             res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1111         end
1112         dx = dx + (r2l and 0 or wd)
1113     end
1114 elseif curr.id == node.id"glue" then
1115     local width = node.effective_glue(curr, box)/factor
1116     dx = dx - (r2l and width or 0)
1117     if curr.leader then
1118         local curr, kind = curr.leader, curr.subtype
1119         if curr.id == node.id"rule" then
1120             local wd, ht, dp = getrulemetric(box, curr, true)
1121             local hd = ht + dp
1122             if hd ~= 0 then
1123                 wd = width
1124                 if wd ~= 0 and curr.subtype == 0 then
1125                     res[#res+1] = rulefmt:format(#res+1, wd, hd, xshift+dx+wd/2, yshift+(ht-dp)/2)
1126                 end
1127             end
1128         elseif curr.head then
1129             local wd = curr.width/factor
1130             if wd <= width then
1131                 local dx = r2l and dx+width or dx
1132                 local n, ix = 0, 0

```

```

1133     if kind == 100 or kind == 103 then -- todo: gleaders
1134         local adx = abs(dx-dirs[1].dx)
1135         local ndx = math.ceil(adx / wd) * wd
1136         local diff = ndx - adx
1137         n = (width-diff) // wd
1138         dx = dx + (r2l and -diff-wd or diff)
1139     else
1140         n = width // wd
1141         if kind == 101 then
1142             local side = width % wd / 2
1143             dx = dx + (r2l and -side-wd or side)
1144         elseif kind == 102 then
1145             ix = width % wd / (n+1)
1146             dx = dx + (r2l and -ix-wd or ix)
1147         end
1148     end
1149     wd = r2l and -wd or wd
1150     ix = r2l and -ix or ix
1151     local func = curr.id == node.id"hlist" and outline_horz or outline_vert
1152     for i=1,n do
1153         res = func(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1154         dx = dx + wd + ix
1155     end
1156 end
1157 end
1158 end
1159 dx = dx + (r2l and 0 or width)
1160 elseif curr.id == node.id"kern" then
1161     dx = dx + curr.kern/factor * (r2l and -1 or 1)
1162 elseif curr.id == node.id"math" then
1163     dx = dx + curr.surround/factor * (r2l and -1 or 1)
1164 elseif curr.id == node.id"vlist" then
1165     dx = dx - (r2l and curr.width/factor or 0)
1166     res = outline_vert(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1167     dx = dx + (r2l and 0 or curr.width/factor)
1168 elseif curr.id == node.id"hlist" then
1169     dx = dx - (r2l and curr.width/factor or 0)
1170     res = outline_horz(res, curr, curr.head, xshift+dx, yshift-curr.shift/factor)
1171     dx = dx + (r2l and 0 or curr.width/factor)
1172 end
1173 curr = node.getnext(curr)
1174 end
1175 return res
1176 end
1177 function luamplib.outlinetext (text)
1178     local fmt = process_tex_text(text)
1179     local id = tonumber(fmt:match"mplibtexboxid=(%d+):")
1180     local box = texgetbox(id)
1181     local res = outline_horz({ }, box, box.head, 0, 0)
1182     if #res == 0 then res = { "mpliboutlinepic[1]:=image();" } end
1183     return tableconcat(res) .. format("mpliboutlinenum:=%i;", #res)
1184 end
1185

```

Our METAPOST preambles

```
1186 luamplib.preambles = {
1187   mplibcode = [[
1188     texscriptmode := 2;
1189     def rawtexttext (expr t) = runscript("luamplibtext{"&t&}") enddef;
1190     def mplibcolor (expr t) = runscript("luamplibcolor{"&t&}") enddef;
1191     def mplibdimen (expr t) = runscript("luamplibdimen{"&t&}") enddef;
1192     def VerbatimTeX (expr t) = runscript("luamplibverbtex{"&t&}") enddef;
1193     if known context_mlib:
1194       defaultfont := "cmtt10";
1195       let infont = normalinfont;
1196       let fontsize = normalfontsize;
1197       vardef thelabel@#(expr p,z) =
1198         if string p :
1199           thelabel@#(p infont defaultfont scaled defaultscale,z)
1200         else :
1201           p shifted (z + labeloffset*mfun_laboff@# -
1202             (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
1203               (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
1204         fi
1205       enddef;
1206     else:
1207       vardef texttext@# (text t) = rawtexttext (t) enddef;
1208     def message expr t =
1209       if string t: runscript("mp.report[="&t&"]=") else: errmessage "Not a string" fi
1210     enddef;
1211   fi
1212   def resolvedcolor(expr s) =
1213     runscript("return luamplib.shadecolor('"&s & "')")
1214   enddef;
1215   def colordecimals primary c =
1216     if cmykcolor c:
1217       decimal cyanpart c & ":" & decimal magentapart c & ":" &
1218       decimal yellowpart c & ":" & decimal blackpart c
1219     elseif rgbcolor c:
1220       decimal redpart c & ":" & decimal greenpart c & ":" & decimal bluepart c
1221     elseif string c:
1222       if known graphicstpic: c else: colordecimals resolvedcolor(c) fi
1223     else:
1224       decimal c
1225     fi
1226   enddef;
1227   def externalfigure primary filename =
1228     draw rawtexttext("\includegraphics{"& filename &}")
1229   enddef;
1230   def TEX = texttext enddef;
1231   def mplibtexcolor primary c =
1232     runscript("return luamplib.gettexcolor('"&c & "')")
1233   enddef;
1234   def mplibrgbtexcolor primary c =
1235     runscript("return luamplib.gettexcolor('"&c & "','rgb')")
1236   enddef;
1237   def mplibgraphicstext primary t =
1238     begingroup;
```



```

1239 mplibgraphictext_ (t)
1240 enddef;
1241 def mplibgraphictext_ (expr t) text rest =
1242   save fakebold, scale, fillcolor, drawcolor, withfillcolor, withdrawcolor,
1243   fb, fc, dc, graphictextpic;
1244   picture graphictextpic; graphictextpic := nullpicture;
1245   numeric fb; string fc, dc; fb:=2; fc:="white"; dc:="black";
1246   let scale = scaled;
1247   def fakebold primary c = hide(fb:=c;) enddef;
1248   def fillcolor primary c = hide(fc:=colordecimals c;) enddef;
1249   def drawcolor primary c = hide(dc:=colordecimals c;) enddef;
1250   let withfillcolor = fillcolor; let withdrawcolor = drawcolor;
1251   addto graphictextpic doublepath origin rest; graphictextpic:=nullpicture;
1252   def fakebold primary c = enddef;
1253   let fillcolor = fakebold; let drawcolor = fakebold;
1254   let withfillcolor = fillcolor; let withdrawcolor = drawcolor;
1255   image(draw runscript("return luamplib.graphictext([===["&t&"]===],"
1256     & decimal fb &","& fc &","& dc &")) rest;)
1257   endgroup;
1258 enddef;
1259 def mplibglyph expr c of f =
1260   runscript (
1261     "return luamplib.glyph('"
1262     & if numeric f: decimal fi f
1263     & "'',"
1264     & if numeric c: decimal fi c
1265     & "'')"
1266   )
1267 enddef;
1268 def mplibdrawglyph expr g =
1269   draw image(
1270     save i; numeric i; i:=0;
1271     for item within g:
1272       i := i+1;
1273       fill pathpart item
1274       if i < length g: withpostscript "collect" fi;
1275     endfor
1276   )
1277 enddef;
1278 def mplib_do_outline_text_set_b (text f) (text d) text r =
1279   def mplib_do_outline_options_f = f enddef;
1280   def mplib_do_outline_options_d = d enddef;
1281   def mplib_do_outline_options_r = r enddef;
1282 enddef;
1283 def mplib_do_outline_text_set_f (text f) text r =
1284   def mplib_do_outline_options_f = f enddef;
1285   def mplib_do_outline_options_r = r enddef;
1286 enddef;
1287 def mplib_do_outline_text_set_u (text f) text r =
1288   def mplib_do_outline_options_f = f enddef;
1289 enddef;
1290 def mplib_do_outline_text_set_d (text d) text r =
1291   def mplib_do_outline_options_d = d enddef;
1292   def mplib_do_outline_options_r = r enddef;

```

```

1293 enddef;
1294 def mplib_do_outline_text_set_r (text d) (text f) text r =
1295   def mplib_do_outline_options_d = d enddef;
1296   def mplib_do_outline_options_f = f enddef;
1297   def mplib_do_outline_options_r = r enddef;
1298 enddef;
1299 def mplib_do_outline_text_set_n text r =
1300   def mplib_do_outline_options_r = r enddef;
1301 enddef;
1302 def mplib_do_outline_text_set_p = enddef;
1303 def mplib_fill_outline_text =
1304   for n=1 upto mpliboutlinenum:
1305     i:=0;
1306     for item within mpliboutlinepic[n]:
1307       i:=i+1;
1308       fill pathpart item mplib_do_outline_options_f withpen pencircle scaled 0
1309       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]): withpostscript "collect"; fi
1310     endfor
1311   endfor
1312 enddef;
1313 def mplib_draw_outline_text =
1314   for n=1 upto mpliboutlinenum:
1315     for item within mpliboutlinepic[n]:
1316       draw pathpart item mplib_do_outline_options_d;
1317     endfor
1318   endfor
1319 enddef;
1320 def mplib_filldraw_outline_text =
1321   for n=1 upto mpliboutlinenum:
1322     i:=0;
1323     for item within mpliboutlinepic[n]:
1324       i:=i+1;
1325       if (n<mpliboutlinenum) or (i<length mpliboutlinepic[n]):
1326         fill pathpart item mplib_do_outline_options_f withpostscript "collect";
1327       else:
1328         draw pathpart item mplib_do_outline_options_f withpostscript "both";
1329       fi
1330     endfor
1331   endfor
1332 enddef;
1333 vardef mpliboutlinetext@# (expr t) text rest =
1334   save kind; string kind; kind := str @#;
1335   save i; numeric i;
1336   picture mpliboutlinepic[]; numeric mpliboutlinenum;
1337   def mplib_do_outline_options_d = enddef;
1338   def mplib_do_outline_options_f = enddef;
1339   def mplib_do_outline_options_r = enddef;
1340   runscript("return luamplib.outlinetext[==["&t&"]===");
1341   image ( addto currentpicture also image (
1342     if kind = "f":
1343       mplib_do_outline_text_set_f rest;
1344       mplib_fill_outline_text;
1345     elseif kind = "d":
1346       mplib_do_outline_text_set_d rest;

```

```

1347     mplib_draw_outline_text;
1348   elseif kind = "b":
1349     mplib_do_outline_text_set_b rest;
1350     mplib_fill_outline_text;
1351     mplib_draw_outline_text;
1352   elseif kind = "u":
1353     mplib_do_outline_text_set_u rest;
1354     mplib_filldraw_outline_text;
1355   elseif kind = "r":
1356     mplib_do_outline_text_set_r rest;
1357     mplib_draw_outline_text;
1358     mplib_fill_outline_text;
1359   elseif kind = "p":
1360     mplib_do_outline_text_set_p;
1361     mplib_draw_outline_text;
1362   else:
1363     mplib_do_outline_text_set_n rest;
1364     mplib_fill_outline_text;
1365   fi;
1366 ) mplib_do_outline_options_r; )
1367 endif;
1368 primarydef t withpattern p =
1369   image(
1370     if cycle t:
1371       fill
1372     else:
1373       draw
1374     fi
1375     t withprescript "mplibpattern=" & if numeric p: decimal fi p; )
1376 endif;
1377 vardef mplibtransformmatrix (text e) =
1378   save t; transform t;
1379   t = identity e;
1380   runscript("luamplib.transformmatrix = {"
1381     & decimal xpart t & ","
1382     & decimal ypart t & ","
1383     & decimal xpart t & ","
1384     & decimal ypart t & ","
1385     & decimal xpart t & ","
1386     & decimal ypart t & ","
1387     & "}");
1388 endif;
1389 primarydef p withfademethod s =
1390   if picture p:
1391     image(
1392       draw p;
1393       draw center p withprescript "mplibfadestate=stop";
1394     )
1395   else:
1396     p withprescript "mplibfadestate=stop"
1397   fi
1398   withprescript "mplibfadetype=" & s
1399   withprescript "mplibfadebbox=" &
1400     decimal (xpart llcorner p -1/4) & ":" &

```

```

1401     decimal (ypart llcorner p -1/4) & ":" &
1402     decimal (xpart urcorner p +1/4) & ":" &
1403     decimal (ypart urcorner p +1/4)
1404 enddef;
1405 def withfadeopacity (expr a,b) =
1406   withprescript "mplibfadeopacity=" &
1407     decimal a & ":" &
1408     decimal b
1409 enddef;
1410 def withfadevector (expr a,b) =
1411   withprescript "mplibfadevector=" &
1412     decimal xpart a & ":" &
1413     decimal ypart a & ":" &
1414     decimal xpart b & ":" &
1415     decimal ypart b
1416 enddef;
1417 let withfadecenter = withfadevector;
1418 def withfaderadius (expr a,b) =
1419   withprescript "mplibfaderadius=" &
1420     decimal a & ":" &
1421     decimal b
1422 enddef;
1423 def withfadebbox (expr a,b) =
1424   withprescript "mplibfadebbox=" &
1425     decimal xpart a & ":" &
1426     decimal ypart a & ":" &
1427     decimal xpart b & ":" &
1428     decimal ypart b
1429 enddef;
1430 primarydef p asgroup s =
1431   image(
1432     draw center p
1433     withprescript "mplibgroupbbox=" &
1434       decimal (xpart llcorner p -1/4) & ":" &
1435       decimal (ypart llcorner p -1/4) & ":" &
1436       decimal (xpart urcorner p +1/4) & ":" &
1437       decimal (ypart urcorner p +1/4)
1438     withprescript "gr_state=start"
1439     withprescript "gr_type=" & s;
1440     draw p;
1441     draw center p withprescript "gr_state=stop";
1442   )
1443 enddef;
1444 def withgroupbbox (expr a,b) =
1445   withprescript "mplibgroupbbox=" &
1446     decimal xpart a & ":" &
1447     decimal ypart a & ":" &
1448     decimal xpart b & ":" &
1449     decimal ypart b
1450 enddef;
1451 def withgroupname expr s =
1452   withprescript "mplibgroupname=" & s
1453 enddef;
1454 def usemplibgroup primary s =

```

```

1455 draw maketext("\mplibnoforcehmode\csname luamplib.group." & s & "\endcsname")
1456   shifted runscript("return luamplib.trgroupshifts['" & s & "']")
1457 enddef;
1458 ]],
1459 legacyverbatimtex = [[
1460 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
1461 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
1462 let VerbatimTeX = specialVerbatimTeX;
1463 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
1464   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
1465 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
1466   "runscript(" &ditto&
1467   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
1468   "luamplib.in_the_fig=false" &ditto& ");";
1469 ]],
1470 texttextlabel = [[
1471 let luampliboriginalinfont = infont;
1472 primarydef s infont f =
1473   if (s < char 32)
1474     or (s = char 35) % #
1475     or (s = char 36) % $
1476     or (s = char 37) % %
1477     or (s = char 38) % &
1478     or (s = char 92) % \
1479     or (s = char 94) % ^
1480     or (s = char 95) % _
1481     or (s = char 123) % {
1482     or (s = char 125) % }
1483     or (s = char 126) % ~
1484     or (s = char 127) :
1485     s luampliboriginalinfont f
1486   else :
1487     rawtexttext(s)
1488   fi
1489 enddef;
1490 def fontsize expr f =
1491   begingroup
1492   save size; numeric size;
1493   size := mplibdimen("1em");
1494   if size = 0: 10pt else: size fi
1495   endgroup
1496 enddef;
1497 ]],
1498 }
1499

```

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

```

1500 luamplib.verbatiminput = false

```

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

```

1501 local function protect_expansion (str)
1502   if str then
1503     str = str:gsub("\\", "!!!Control!!!")
1504           :gsub("%%", "!!!Comment!!!")
1505           :gsub("#", "!!!HashSign!!!")

```

```

1506         :gsub("{", "!!!LBrace!!!")
1507         :gsub("}", "!!!RBrace!!!")
1508     return format("\\unexpanded{%s}",str)
1509 end
1510 end
1511 local function unprotect_expansion (str)
1512     if str then
1513         return str:gsub("!!!Control!!!", "\\")
1514             :gsub("!!!Comment!!!", "%")
1515             :gsub("!!!HashSign!!!", "#")
1516             :gsub("!!!LBrace!!!", "{")
1517             :gsub("!!!RBrace!!!", "}")
1518     end
1519 end
1520 luamplib.everymplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1521 luamplib.everyendmplib = setmetatable({ [""] = "" },{ __index = function(t) return t[""] end })
1522 function luamplib.process_mplibcode (data, instancename)
1523     texboxes.localid = 4096

```

This is needed for legacy behavior

```

1524 if luamplib.legacyverbatim then
1525     luamplib.figid, tex_code_pre_mplib = 1, {}
1526 end
1527 local everymplib = luamplib.everymplib[instancename]
1528 local everyendmplib = luamplib.everyendmplib[instancename]
1529 data = format("\\n%s\\n%s\\n%s\\n",everymplib, data, everyendmplib)
1530 :gsub("\\r", "\\n")

```

These five lines are needed for mplibverbatim mode.

```

1531 if luamplib.verbatiminput then
1532     data = data:gsub("\\mpcolor%+{.-%b{}}", "mplibcolor(\\%1\\)")
1533     :gsub("\\mpdim%+{.-%b{}}", "mplibdimen(\\%1\\)")
1534     :gsub("\\mpdim%+{.-%a+}", "mplibdimen(\\%1\\)")
1535     :gsub(btex_etex, "btex %1 etex ")
1536     :gsub(verbatimetex_etex, "verbatimetex %1 etex;")

```

If not mplibverbatim, expand mplibcode data, so that users can use \TeX codes in it. It has turned out that no comment sign is allowed.

```

1537 else
1538     data = data:gsub(btex_etex, function(str)
1539         return format("btex %s etex ", protect_expansion(str)) -- space
1540     end)
1541     :gsub(verbatimetex_etex, function(str)
1542         return format("verbatimetex %s etex;", protect_expansion(str)) -- semicolon
1543     end)
1544     :gsub("\\.-\\", protect_expansion)
1545     :gsub("\\%%", "\\0PerCent\0")
1546     :gsub("%%.-\\n", "\\n")
1547     :gsub("%zPerCentz", "\\%z")
1548     run_tex_code(format("\\mplibtmptoks\\expandafter{\\expanded{%s}}",data))
1549     data = texgettoks"mplibtmptoks"

```

Next line to address issue #55

```

1550     :gsub("##", "#")
1551     :gsub("\\.-\\", unprotect_expansion)
1552     :gsub(btex_etex, function(str)

```

```

1553     return format("btex %s etex", unprotect_expansion(str))
1554   end)
1555   :gsub(verbatimtex_etex, function(str)
1556     return format("verbatimtex %s etex", unprotect_expansion(str))
1557   end)
1558 end
1559 process(data, instancename)
1560 end
1561

```

For parsing prescript materials.

```

1562 local function script2table(s)
1563   local t = {}
1564   for _,i in ipairs(s:explode("\13+")) do
1565     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
1566     if k and v and k ~= "" and not t[k] then
1567       t[k] = v
1568     end
1569   end
1570   return t
1571 end
1572

```

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.

```

1573 local figcontents = { post = { } }
1574 local function put2output(a,...)
1575   figcontents[#figcontents+1] = type(a) == "string" and format(a,...) or a
1576 end
1577 local function pdf_startfigure(n,llx,lly,urx,ury)
1578   put2output("\mplibstarttoPDF{%f}{%f}{%f}{%f}", llx, lly, urx, ury)
1579 end
1580 local function pdf_stopfigure()
1581   put2output("\mplibstoptoPDF")
1582 end

```

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

```

1583 local function pdf_literalcode (...)
1584   put2output{ -2, format(...) :gsub(decimals,rmzeros) }
1585 end
1586 local start_pdf_code = pdfmode
1587 and function() pdf_literalcode"q" end
1588 or function() put2output"\special{pdf:bcontent}" end
1589 local stop_pdf_code = pdfmode
1590 and function() pdf_literalcode"Q" end
1591 or function() put2output"\special{pdf:econtent}" end
1592

```

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

```

1593 local function put_tex_box(es (object,prescript)
1594   local box = prescript.mplibtexboxid:explode":"
1595   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
1596   if n and tw and th then

```

```

1597 local op = object.path
1598 local first, second, fourth = op[1], op[2], op[4]
1599 local tx, ty = first.x_coord, first.y_coord
1600 local sx, rx, ry, sy = 1, 0, 0, 1
1601 if tw ~= 0 then
1602     sx = (second.x_coord - tx)/tw
1603     rx = (second.y_coord - ty)/tw
1604     if sx == 0 then sx = 0.00001 end
1605 end
1606 if th ~= 0 then
1607     sy = (fourth.y_coord - ty)/th
1608     ry = (fourth.x_coord - tx)/th
1609     if sy == 0 then sy = 0.00001 end
1610 end
1611 start_pdf_code()
1612 pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
1613 put2output("\\mplibputtextbox{%i}",n)
1614 stop_pdf_code()
1615 end
1616 end
1617

```

Colors

```

1618 local prev_override_color
1619 local function do_preobj_CR(object,prescript)
1620 if object.postscript == "collect" then return end
1621 local override = prescript and prescript.mpliboverridecolor
1622 if override then
1623     if pdfmode then
1624         pdf_literalcode(override)
1625         override = nil
1626     else
1627         put2output("\\special{%s}",override)
1628         prev_override_color = override
1629     end
1630 else
1631     local cs = object.color
1632     if cs and #cs > 0 then
1633         pdf_literalcode(luamplib.colorconverter(cs))
1634         prev_override_color = nil
1635     elseif not pdfmode then
1636         override = prev_override_color
1637         if override then
1638             put2output("\\special{%s}",override)
1639         end
1640     end
1641 end
1642 return override
1643 end
1644

```

For transparency and shading

```

1645 local pdfmanagement = is_defined'pdfmanagement_add:nnn'
1646 local pdfobjs, pdfetcs = {}, {}
1647 pdfetcs.pgftxtgs = "pgf@sys@addpdfresource@extgs@plain"

```



```

1648 pdfetcs.pgfpattern = "pgf@sys@addpdfresource@patterns@plain"
1649 pdfetcs.pgfcolorspace = "pgf@sys@addpdfresource@colorspaces@plain"
1650 local function update_pdfobjs (os, stream)
1651   local key = os
1652   if stream then key = key..stream end
1653   local on = pdfobjs[key]
1654   if on then
1655     return on,false
1656   end
1657   if pdfmode then
1658     if stream then
1659       on = pdf.immediateobj("stream",stream,os)
1660     else
1661       on = pdf.immediateobj(os)
1662     end
1663   else
1664     on = pdfetcs.cnt or 1
1665     if stream then
1666       texsprintf(format("\\special{pdf:stream @mplibpdfobj%s (%s) <<%s>>}",on,stream,os))
1667     else
1668       texsprintf(format("\\special{pdf:obj @mplibpdfobj%s %s}",on,os))
1669     end
1670     pdfetcs.cnt = on + 1
1671   end
1672   pdfobjs[key] = on
1673   return on,true
1674 end
1675 pdfetcs.resfmt = pdfmode and "%s 0 R" or "@mplibpdfobj%s"
1676 if pdfmode then
1677   pdfetcs.getpagers = pdf.getpagersources or function() return pdf.pagersources end
1678   local getpagers = pdfetcs.getpagers
1679   local setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
1680   local initialize_resources = function (name)
1681     local tabname = format("%s_res",name)
1682     pdfetcs[tabname] = { }
1683     if luatexbase.callbacktypes.finish_pdffile then -- ltuatex
1684       local obj = pdf.reserveobj()
1685       setpagers(format("%s/%s %i 0 R", getpagers() or "", name, obj))
1686       luatexbase.add_to_callback("finish_pdffile", function()
1687         pdf.immediateobj(obj, format("<<%s>>", tableconcat(pdfetcs[tabname])))
1688       end,
1689         format("luamplib.%s.finish_pdffile",name))
1690     end
1691   end
1692   pdfetcs.fallback_update_resources = function (name, res)
1693     local tabname = format("%s_res",name)
1694     if not pdfetcs[tabname] then
1695       initialize_resources(name)
1696     end
1697     if luatexbase.callbacktypes.finish_pdffile then
1698       local t = pdfetcs[tabname]
1699       t[#t+1] = res
1700     else
1701       local tpr, n = getpagers() or "", 0

```

```

1702     tpr, n = tpr:gsub(format("/%s<<",name), "%1".res)
1703     if n == 0 then
1704         tpr = format("%s/%s<<%s>>", tpr, name, res)
1705     end
1706     setpageres(tpr)
1707 end
1708 end
1709 else
1710 texsprint {
1711     "\\luamplibatfirstshipout{",
1712     "\\special{pdf:obj @MPLibTr<<>>}",
1713     "\\special{pdf:obj @MPLibSh<<>>}",
1714     "\\special{pdf:obj @MPLibCS<<>>}",
1715     "\\special{pdf:obj @MPLibPt<<>>}",
1716 }
1717 pdfetcs.resadded = { }
1718 pdfetcs.fallback_update_resources = function (name,res,obj)
1719     texsprint{"\\special{pdf:put ", obj, " <<", res, ">>}"}
1720     if not pdfetcs.resadded[name] then
1721         texsprint{"\\luamplibateveryshipout{\\special{pdf:put @resources <</", name, " ", obj, ">>}"}
1722         pdfetcs.resadded[name] = obj
1723     end
1724 end
1725 end
1726

```

Transparency

```

1727 local transparency_modes = { [0] = "Normal",
1728     "Normal",     "Multiply",     "Screen",     "Overlay",
1729     "SoftLight",  "HardLight",   "ColorDodge", "ColorBurn",
1730     "Darken",     "Lighten",     "Difference",  "Exclusion",
1731     "Hue",        "Saturation",  "Color",      "Luminosity",
1732     "Compatible",
1733 }
1734 local function add_extgs_resources (on, new)
1735     local key = format("MPLibTr%s", on)
1736     if new then
1737         local val = format(pdfetcs.resfmt, on)
1738         if pdfmanagement then
1739             texsprint {
1740                 "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ExtGState}{", key, "}{" , val, "}"
1741             }
1742         else
1743             local tr = format("/%s %s", key, val)
1744             if is_defined(pdfetcs.pgfextgs) then
1745                 texsprint { "\\csname ", pdfetcs.pgfextgs, "\\endcsname{", tr, "}" }
1746             elseif is_defined"TRP@list" then
1747                 texsprint(catat11,{
1748                     [[\if@files\immediate\write\@auxout{]],
1749                     [[\string\g@addto@macro\string\TRP@list{]],
1750                     tr,
1751                     [[]}\fi]],
1752                 })
1753             if not get_macro"TRP@list":find(tr) then
1754                 texsprint(catat11,[[\global\TRP@reruntrue]])

```

```

1755     end
1756   else
1757     pdfetcs.fallback_update_resources("ExtGState",tr,"@MPlibTr")
1758   end
1759 end
1760 end
1761 return key
1762 end
1763 local function do_preobj_TR(object,prescript)
1764 if object.postscript == "collect" then return end
1765 local opaq = prescript and prescript.tr_transparency
1766 if opaq then
1767   local key, on, os, new
1768   local mode = prescript.tr_alternative or 1
1769   mode = transparency_modes[tonumber(mode)] or mode
1770   opaq = format("%.3f", opaq) :gsub(decimals,rmzeros)
1771   for i,v in ipairs{ {mode,opaq},{ "Normal",1} } do
1772     os = format("<</BM/%s/ca %s/CA %s/AIS false>>",v[1],v[2],v[2])
1773     on, new = update_pdfobjs(os)
1774     key = add_extgs_resources(on,new)
1775     if i == 1 then
1776       pdf_literalcode("/%s gs",key)
1777     else
1778       return format("/%s gs",key)
1779     end
1780   end
1781 end
1782 end
1783

```

Shading with *metafun* format.

```

1784 local function sh_pdfpageresources(shtype,domain,colorspace,ca,cb,coordinates,steps,fractions)
1785 for _,v in ipairs{ca,cb} do
1786   for i,vv in ipairs(v) do
1787     for ii,vvv in ipairs(vv) do
1788       v[i][ii] = tonumber(vvv) and format("%.3f",vvv) or vvv
1789     end
1790   end
1791 end
1792 local fun2fmt,os = "<</FunctionType 2/Domain[%s]/C0[%s]/C1[%s]/N 1>>"
1793 if steps > 1 then
1794   local list,bounds,encode = { },{ },{ }
1795   for i=1,steps do
1796     if i < steps then
1797       bounds[i] = format("%.3f", fractions[i] or 1)
1798     end
1799     encode[2*i-1] = 0
1800     encode[2*i] = 1
1801     os = fun2fmt:format(domain,tableconcat(ca[i],' '),tableconcat(cb[i],' '))
1802     :gsub(decimals,rmzeros)
1803     list[i] = format(pdfetcs.resfmt, update_pdfobjs(os))
1804   end
1805   os = tableconcat {
1806     "<</FunctionType 3",
1807     format("/Bounds[%s]", tableconcat(bounds,' ')),

```

```

1808     format("/Encode[%s]", tableconcat(encode, ' ')),
1809     format("/Functions[%s]", tableconcat(list, ' ')),
1810     format("/Domain[%s]>>", domain),
1811   } :gsub(decimals,rmzeros)
1812 else
1813   os = fun2fmt:format(domain,tableconcat(ca[1], ' '),tableconcat(cb[1], ' '))
1814   :gsub(decimals,rmzeros)
1815 end
1816 local objref = format(pdfetcs.resfmt, update_pdfobjs(os))
1817 os = tableconcat {
1818   format("<</ShadingType %i", shtype),
1819   format("/ColorSpace %s", colorspace),
1820   format("/Function %s", objref),
1821   format("/Coords[%s]", coordinates),
1822   "/Extend[true true]/AntiAlias true>>",
1823 } :gsub(decimals,rmzeros)
1824 local on, new = update_pdfobjs(os)
1825 if new then
1826   local key, val = format("MPLibSh%s", on), format(pdfetcs.resfmt, on)
1827   if pdfmanagement then
1828     texsprintf {
1829       "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Shading}{", key, "}{" , val, "}"
1830     }
1831   else
1832     local res = format("/%s %s", key, val)
1833     pdfetcs.fallback_update_resources("Shading",res,"@MPLibSh")
1834   end
1835 end
1836 return on
1837 end
1838 local function color_normalize(ca,cb)
1839   if #cb == 1 then
1840     if #ca == 4 then
1841       cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
1842     else -- #ca = 3
1843       cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
1844     end
1845   elseif #cb == 3 then -- #ca == 4
1846     cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
1847   end
1848 end
1849 pdfetcs.clrspcs = setmetatable({ }, { __index = function(t,names)
1850   run_tex_code({
1851     [[\color_model_new:nnn]],
1852     format("{mplibcolorspace_%s}", names:gsub(",","_")),
1853     format("{DeviceN}{names={%s}}", names),
1854     [[\edef\mplib@tempa{\pdf_object_ref_last:}]],
1855   }, ccexplat)
1856   local colorspace = get_macro'mplib@tempa'
1857   t[names] = colorspace
1858   return colorspace
1859 end })
1860 local function do_preobj_SH(object,prescript)
1861   local shade_no

```

```

1862 local sh_type = prescript and prescript.sh_type
1863 if not sh_type then
1864     return
1865 else
1866     local domain = prescript.sh_domain or "0 1"
1867     local centera = (prescript.sh_center_a or "0 0"):explode()
1868     local centerb = (prescript.sh_center_b or "0 0"):explode()
1869     local transform = prescript.sh_transform == "yes"
1870     local sx,sy,sr,dx,dy = 1,1,1,0,0
1871     if transform then
1872         local first = (prescript.sh_first or "0 0"):explode()
1873         local setx = (prescript.sh_set_x or "0 0"):explode()
1874         local sety = (prescript.sh_set_y or "0 0"):explode()
1875         local x,y = tonumber(setx[1]) or 0, tonumber(sety[1]) or 0
1876         if x ~= 0 and y ~= 0 then
1877             local path = object.path
1878             local path1x = path[1].x_coord
1879             local path1y = path[1].y_coord
1880             local path2x = path[x].x_coord
1881             local path2y = path[y].y_coord
1882             local dxa = path2x - path1x
1883             local dya = path2y - path1y
1884             local dxb = setx[2] - first[1]
1885             local dyb = sety[2] - first[2]
1886             if dxa ~= 0 and dya ~= 0 and dxb ~= 0 and dyb ~= 0 then
1887                 sx = dxa / dxb ; if sx < 0 then sx = - sx end
1888                 sy = dya / dyb ; if sy < 0 then sy = - sy end
1889                 sr = math.sqrt(sx^2 + sy^2)
1890                 dx = path1x - sx*first[1]
1891                 dy = path1y - sy*first[2]
1892             end
1893         end
1894     end
1895     local ca, cb, colorspace, steps, fractions
1896     ca = { (prescript.sh_color_a_1 or prescript.sh_color_a or "0"):explode:" }
1897     cb = { (prescript.sh_color_b_1 or prescript.sh_color_b or "1"):explode:" }
1898     steps = tonumber(prescript.sh_step) or 1
1899     if steps > 1 then
1900         fractions = { prescript.sh_fraction_1 or 0 }
1901         for i=2,steps do
1902             fractions[i] = prescript[format("sh_fraction_%i",i)] or (i/steps)
1903             ca[i] = (prescript[format("sh_color_a_%i",i)] or "0"):explode:"
1904             cb[i] = (prescript[format("sh_color_b_%i",i)] or "1"):explode:"
1905         end
1906     end
1907     if prescript.mplib_spotcolor then
1908         ca, cb = { }, { }
1909         local names, pos, objref = { }, -1, ""
1910         local script = object.prescript:explode"\13+"
1911         for i=#script,1,-1 do
1912             if script[i]:find"mplib_spotcolor" then
1913                 local t, name, value = script[i]:explode"="[2]:explode:"
1914                 value, objref, name = t[1], t[2], t[3]
1915                 if not names[name] then

```

```

1916         pos = pos+1
1917         names[name] = pos
1918         names[#names+1] = name
1919     end
1920     t = { }
1921     for j=1,names[name] do t[#t+1] = 0 end
1922     t[#t+1] = value
1923     tableinsert(#ca == #cb and ca or cb, t)
1924 end
1925 end
1926 for _,t in ipairs{ca,cb} do
1927     for _,tt in ipairs(t) do
1928         for i=1,#names-#tt do tt[#tt+1] = 0 end
1929     end
1930 end
1931 if #names == 1 then
1932     colorspace = objref
1933 else
1934     colorspace = pdfetcs.clrspcs[ tableconcat(names,",") ]
1935 end
1936 else
1937     local model = 0
1938     for _,t in ipairs{ca,cb} do
1939         for _,tt in ipairs(t) do
1940             model = model > #tt and model or #tt
1941         end
1942     end
1943     for _,t in ipairs{ca,cb} do
1944         for _,tt in ipairs(t) do
1945             if #tt < model then
1946                 color_normalize(model == 4 and {1,1,1,1} or {1,1,1},tt)
1947             end
1948         end
1949     end
1950     colorspace = model == 4 and "/DeviceCMYK"
1951                 or model == 3 and "/DeviceRGB"
1952                 or model == 1 and "/DeviceGray"
1953                 or err"unknown color model"
1954 end
1955 if sh_type == "linear" then
1956     local coordinates = format("%f %f %f %f",
1957         dx + sx*centera[1], dy + sy*centera[2],
1958         dx + sx*centerb[1], dy + sy*centerb[2])
1959     shade_no = sh_pdfpageresources(2,domain,colorspace,ca,cb,coordinates,steps,fractions)
1960 elseif sh_type == "circular" then
1961     local factor = prescript.sh_factor or 1
1962     local radiusa = factor * prescript.sh_radius_a
1963     local radiusb = factor * prescript.sh_radius_b
1964     local coordinates = format("%f %f %f %f %f %f",
1965         dx + sx*centera[1], dy + sy*centera[2], sr*radiusa,
1966         dx + sx*centerb[1], dy + sy*centerb[2], sr*radiusb)
1967     shade_no = sh_pdfpageresources(3,domain,colorspace,ca,cb,coordinates,steps,fractions)
1968 else
1969     err"unknown shading type"

```

```

1970   end
1971 end
1972 return shade_no
1973 end
1974
      Patterns
1975 pdfetcs.patterns = { }
1976 local function gather_resources (optres)
1977   local t, do_pattern = { }, not optres
1978   local names = {"ExtGState", "ColorSpace", "Shading"}
1979   if do_pattern then
1980     names[#names+1] = "Pattern"
1981   end
1982   if pdfmode then
1983     if pdfmanagement then
1984       for _,v in ipairs(names) do
1985         local pp = get_macro(format("g__pdfdict_/g__pdf_Core/Page/Resources/%s_prop",v))
1986         if pp and pp:find"__prop_pair" then
1987           t[#t+1] = format("/%s %s 0 R", v, ltx.pdf.object_id("__pdf/Page/Resources/".v))
1988         end
1989       end
1990     else
1991       local res = pdfetcs.getpageres() or ""
1992       run_tex_code[["\mplibtmptoks\expandafter{\the\pdfvariable pageresources}]]
1993       res = res .. texgettoks'mplibtmptoks'
1994       if do_pattern then return res end
1995       res = res:explode"/+"
1996       for _,v in ipairs(res) do
1997         v = v:match"^%s*(.)%s*$"
1998         if not v:find"Pattern" and not optres:find(v) then
1999           t[#t+1] = "/" .. v
2000         end
2001       end
2002     end
2003   else
2004     if pdfmanagement then
2005       for _,v in ipairs(names) do
2006         local pp = get_macro(format("g__pdfdict_/g__pdf_Core/Page/Resources/%s_prop",v))
2007         if pp and pp:find"__prop_pair" then
2008           run_tex_code {
2009             "\mplibtmptoks\expanded{{" ,
2010             format("/%s \\\csname pdf_object_ref:n\\endcsname{__pdf/Page/Resources/%s}",v,v),
2011             "}}",
2012           }
2013           t[#t+1] = texgettoks'mplibtmptoks'
2014         end
2015       end
2016     elseif is_defined(pdfetcs.pgfgextgs) then
2017       run_tex_code {
2018         "\mplibtmptoks\expanded{{" ,
2019         "\\ifpgf@sys@pdf@extgs@exists /ExtGState @pgfgextgs\\fi",
2020         "\\ifpgf@sys@pdf@colorspaces@exists /ColorSpace @pgfcolorspaces\\fi",
2021         do_pattern and "\\ifpgf@sys@pdf@patterns@exists /Pattern @pgfpatterns \\fi" or "",
2022         "}}",

```

```

2023     }, catat11)
2024     t[#t+1] = texgettoks'mplibtmptoks'
2025   else
2026     for _,v in ipairs(names) do
2027       local vv = pdfetcs.resadded[v]
2028       if vv then
2029         t[#t+1] = format("/%s %s", v, vv)
2030       end
2031     end
2032   end
2033 end
2034 return tableconcat(t)
2035 end
2036 function luamplib.registerpattern ( boxid, name, opts )
2037   local box = texgetbox(boxid)
2038   local wd = format("%.3f",box.width/factor)
2039   local hd = format("%.3f", (box.height+box.depth)/factor)
2040   info("w/h/d of pattern '%s': %s 0", name, format("%s %s",wd, hd):gsub(decimals,rmzeros))
2041   if opts.xstep == 0 then opts.xstep = nil end
2042   if opts.ystep == 0 then opts.ystep = nil end
2043   if opts.colored == nil then
2044     opts.colored = opts.coloured
2045     if opts.colored == nil then
2046       opts.colored = true
2047     end
2048   end
2049   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix," ") end
2050   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox," ") end
2051   if opts.matrix and opts.matrix:find"%a" then
2052     local data = format("mplibtransformmatrix(%s);",opts.matrix)
2053     process(data,"@mplibtransformmatrix")
2054     local t = luamplib.transformmatrix
2055     opts.matrix = format("%f %f %f %f", t[1], t[2], t[3], t[4])
2056     opts.xshift = opts.xshift or format("%f",t[5])
2057     opts.yshift = opts.yshift or format("%f",t[6])
2058   end
2059   local attr = {
2060     "/Type/Pattern",
2061     "/PatternType 1",
2062     format("/PaintType %i", opts.colored and 1 or 2),
2063     "/TilingType 2",
2064     format("/XStep %s", opts.xstep or wd),
2065     format("/YStep %s", opts.ystep or hd),
2066     format("/Matrix[%s %s %s]", opts.matrix or "1 0 0 1", opts.xshift or 0, opts.yshift or 0),
2067   }
2068   local optres = opts.resources or ""
2069   optres = optres .. gather_resources(optres)
2070   local patterns = pdfetcs.patterns
2071   if pdfmode then
2072     if opts.bbox then
2073       attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2074     end
2075     attr = tableconcat(attr) :gsub(decimals,rmzeros)
2076     local index = tex.saveboxresource(boxid, attr, optres, true, opts.bbox and 4 or 1)

```



```

2077 patterns[name] = { id = index, colored = opts.colored }
2078 else
2079 local cnt = #patterns + 1
2080 local objname = "@mplibpattern" .. cnt
2081 local metric = format("bbox %s", opts.bbox or format("0 0 %s %s",wd,hd))
2082 texsprintf {
2083   "\\expandafter\\newbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2084   "\\global\\setbox\\csname luamplib.patternbox.", cnt, "\\endcsname",
2085   "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2086   "\\special{pdf:bcontent}",
2087   "\\special{pdf:bxobj ", objname, " ", metric, "}",
2088   "\\raise\\dp\\csname luamplib.patternbox.", cnt, "\\endcsname",
2089   "\\box\\csname luamplib.patternbox.", cnt, "\\endcsname",
2090   "\\special{pdf:put @resources <<", optres, ">>}",
2091   "\\special{pdf:exobj <<", tableconcat(attr), ">>}",
2092   "\\special{pdf:econtent}}",
2093 }
2094 patterns[cnt] = objname
2095 patterns[name] = { id = cnt, colored = opts.colored }
2096 end
2097 end
2098 local function pattern_colorspace (cs)
2099 local on, new = update_pdfobjs(format("[/Pattern %s]", cs))
2100 if new then
2101 local key, val = format("MPLibCS%i",on), format(pdfetcs.resfmt,on)
2102 if pdfmanagement then
2103 texsprintf {
2104   "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/ColorSpace}{", key, "}{" , val, "}"
2105 }
2106 else
2107 local res = format("/%s %s", key, val)
2108 if is_defined(pdfetcs.pgfcolorspace) then
2109 texsprintf { "\\csname ", pdfetcs.pgfcolorspace, "\\endcsname{" , res, "}" }
2110 else
2111 pdfetcs.fallback_update_resources("ColorSpace",res,"@MPLibCS")
2112 end
2113 end
2114 end
2115 return on
2116 end
2117 local function do_preobj_PAT(object, prescript)
2118 local name = prescript and prescript.mplibpattern
2119 if not name then return end
2120 local patterns = pdfetcs.patterns
2121 local patt = patterns[name]
2122 local index = patt and patt.id or err("cannot get pattern object '%s'", name)
2123 local key = format("MPLibPt%s",index)
2124 if patt.colored then
2125 pdf_literalcode("/Pattern cs /%s scn", key)
2126 else
2127 local color = prescript.mpliboverridecolor
2128 if not color then
2129 local t = object.color
2130 color = t and #t>0 and luamplib.colorconverter(t)

```

```

2131 end
2132 if not color then return end
2133 local cs
2134 if color:find" cs " or color:find"@pdf.obj" then
2135     local t = color:explode()
2136     if pdfmode then
2137         cs = format("%s 0 R", ltx.pdf.object_id( t[1]:sub(2,-1) ))
2138         color = t[3]
2139     else
2140         cs = t[2]
2141         color = t[3]:match"%[(.+)%"
2142     end
2143 else
2144     local t = colorsplit(color)
2145     cs = #t == 4 and "/DeviceCMYK" or #t == 3 and "/DeviceRGB" or "/DeviceGray"
2146     color = tableconcat(t, " ")
2147 end
2148 pdf_literalcode("/MPLibCS%i cs %s /%s scn", pattern_colorspace(cs), color, key)
2149 end
2150 if not patt.done then
2151     local val = pdfmode and format("%s 0 R",index) or patterns[index]
2152     if pdfmanagement then
2153         texsprintf {
2154             "\\csname pdfmanagement_add:nnn\\endcsname{Page/Resources/Pattern}{", key, "}{" , val, "}"
2155         }
2156     else
2157         local res = format("/%s %s", key, val)
2158         if is_defined(pdfetcs.pgfpattern) then
2159             texsprintf { "\\csname ", pdfetcs.pgfpattern, "\\endcsname{" , res, "}" }
2160         else
2161             pdfetcs.fallback_update_resources("Pattern",res,"@MPLibPt")
2162         end
2163     end
2164 end
2165 patt.done = true
2166 end
2167

```

Fading

```

2168 pdfetcs.fading = { }
2169 local function do_preobj_FADE (object, prescript)
2170     local fd_type = prescript and prescript.mplibfadetype
2171     local fd_stop = prescript and prescript.mplibfadestate
2172     if not fd_type then
2173         return fd_stop -- returns "stop" (if picture) or nil
2174     end
2175     local bbox = prescript.mplibfadebbox:explode:"
2176     local dx, dy = -bbox[1], -bbox[2]
2177     local vec = prescript.mplibfadevector; vec = vec and vec:explode:"
2178     if not vec then
2179         if fd_type == "linear" then
2180             vec = {bbox[1], bbox[2], bbox[3], bbox[2]} -- left to right
2181         else
2182             local centerx, centery = (bbox[1]+bbox[3])/2, (bbox[2]+bbox[4])/2
2183             vec = {centerx, centery, centerx, centery} -- center for both circles

```

```

2184 end
2185 end
2186 local coords = { vec[1]+dx, vec[2]+dy, vec[3]+dx, vec[4]+dy }
2187 if fd_type == "linear" then
2188   coords = format("%f %f %f %f", tableunpack(coords))
2189 elseif fd_type == "circular" then
2190   local width, height = bbox[3]-bbox[1], bbox[4]-bbox[2]
2191   local radius = (prescript.mplibfaderadius or "0"..math.sqrt(width^2+height^2)/2):explode:""
2192   tableinsert(coords, 3, radius[1])
2193   tableinsert(coords, radius[2])
2194   coords = format("%f %f %f %f %f %f", tableunpack(coords))
2195 else
2196   err("unknown fading method '%s'", fd_type)
2197 end
2198 fd_type = fd_type == "linear" and 2 or 3
2199 local opa = (prescript.mplibfadeopacity or "1:0"):explode:""
2200 local on, os, new
2201 on = sh_pdfpageresources(fd_type, "0 1", "/DeviceGray", {{opa[1]}}, {{opa[2]}}, coords, 1)
2202 os = format("<</PatternType 2/Shading %s>>", format(pdfetcs.resfmt, on))
2203 on = update_pdfobjs(os)
2204 bbox = format("0 0 %f %f", bbox[3]+dx, bbox[4]+dy)
2205 local streamtext = format("q /Pattern cs/MPLibFd%s scn %s re f Q", on, bbox)
2206   :gsub(decimals,rmzeros)
2207 os = format("<</Pattern<</MPLibFd%s %s>>>>", on, format(pdfetcs.resfmt, on))
2208 on = update_pdfobjs(os)
2209 local resources = format(pdfetcs.resfmt, on)
2210 on = update_pdfobjs("<</S/Transparency/CS/DeviceGray>>")
2211 local attr = tableconcat{
2212   "/Subtype/Form",
2213   "/BBox[" .. bbox .. "]",
2214   "/Matrix[1 0 0 1 " .. format("%f %f", -dx,-dy) .. "]",
2215   "/Resources " .. resources,
2216   "/Group " .. format(pdfetcs.resfmt, on),
2217 } :gsub(decimals,rmzeros)
2218 on = update_pdfobjs(attr, streamtext)
2219 os = "<</SMask<</S/Luminosity/G " .. format(pdfetcs.resfmt, on) .. ">>>>"
2220 on, new = update_pdfobjs(os)
2221 local key = add_extgs_resources(on,new)
2222 start_pdf_code()
2223 pdf_literalcode("/%s gs", key)
2224 if fd_stop then return "standalone" end
2225 return "start"
2226 end
2227

```

Transparency Group

```

2228 pdfetcs.tr_group = { shifts = { } }
2229 luamplib.trgroupshifts = pdfetcs.tr_group.shifts
2230 local function do_preobj_GRP (object, prescript)
2231   local grstate = prescript and prescript.gr_state
2232   if not grstate then return end
2233   local trgroup = pdfetcs.tr_group
2234   if grstate == "start" then
2235     trgroup.name = prescript.mplibgroupname or "lastmplibgroup"
2236     trgroup.isolated, trgroup.knockout = false, false

```

```

2237 for _,v in ipairs(prescript.gr_type:explode,+) do
2238   trgroup[v] = true
2239 end
2240 trgroup.bbox = prescript.mplibgroupbbox:explode:"
2241 put2output[[\begingroup\setbox\mplibscratchbox\hbox\bgroup]]
2242 elseif grstate == "stop" then
2243   local llx,lly,urx,ury = tableunpack(trgroup.bbox)
2244   put2output(tableconcat{
2245     "\egroup",
2246     format("\wd\mplibscratchbox %fbp", urx-llx),
2247     format("\ht\mplibscratchbox %fbp", ury-lly),
2248     "\dp\mplibscratchbox 0pt",
2249   })
2250   local grattr = format("/Group<</S/Transparency/I %s/K %s>>", trgroup.isolated, trgroup.knockout)
2251   local res = gather_resources()
2252   local bbox = format("%f %f %f %f", llx,lly,urx,ury) :gsub(decimals,rmzeros)
2253   if pdfmode then
2254     put2output(tableconcat{
2255       "\saveboxresource type 2 attr{/Type/XObject/Subtype/Form/FormType 1",
2256       "/BBox[" .. bbox .. "], grattr, "} resources{" .. res .. "}\mplibscratchbox",
2257       [[\setbox\mplibscratchbox\hbox{\useboxresource\lastsavedboxresourceindex}]],
2258       [[\wd\mplibscratchbox 0pt\ht\mplibscratchbox 0pt\dp\mplibscratchbox 0pt]],
2259       [[\box\mplibscratchbox\endgroup]],
2260       "\expandafter\def\csname luamplib.group.", trgroup.name, "\endcsname{",
2261       "\noexpand\mplibstarttoPDF{" .. llx .. "}" .. lly .. "}" .. urx .. "}" .. ury .. "}",
2262       "\useboxresource \the\lastsavedboxresourceindex\noexpand\mplibstoptoPDF",
2263     })
2264   else
2265     trgroup.cnt = (trgroup.cnt or 0) + 1
2266     local objname = format("@mplibtrgr%s", trgroup.cnt)
2267     put2output(tableconcat{
2268       "\special{pdf:boxobj " .. objname .. " bbox " .. bbox .. "}",
2269       "\unhbox\mplibscratchbox",
2270       "\special{pdf:put @resources <<", res, ">>",
2271       "\special{pdf:exobj <<", grattr, ">>",
2272       "\special{pdf:uxobj " .. objname .. "}" .. "\endgroup",
2273     })
2274     token.set_macro("luamplib.group." .. trgroup.name, tableconcat{
2275       "\mplibstarttoPDF{" .. llx .. "}" .. lly .. "}" .. urx .. "}" .. ury .. "}",
2276       "\special{pdf:uxobj " .. objname .. "}" .. "\mplibstoptoPDF",
2277     }, "global")
2278   end
2279   trgroup.shifts[trgroup.name] = { llx, lly }
2280 end
2281 return grstate
2282 end
2283 function luamplib.registergroup (boxid, name, opts)
2284   local box = texgetbox(boxid)
2285   local wd, ht, dp = node.getwhd(box)
2286   local res = (opts.resources or "") .. gather_resources()
2287   local attr = { "/Type/XObject/Subtype/Form/FormType 1" }
2288   if type(opts.matrix) == "table" then opts.matrix = tableconcat(opts.matrix, " ") end
2289   if type(opts.bbox) == "table" then opts.bbox = tableconcat(opts.bbox, " ") end
2290   if opts.matrix and opts.matrix:find"%a" then

```

```

2291 local data = format("mplibtransformmatrix(%s);",opts.matrix)
2292 process(data,"@mplibtransformmatrix")
2293 opts.matrix = format("%f %f %f %f %f %f",tableunpack(luamplib.transformmatrix))
2294 end
2295 local grtype = 3
2296 if opts.bbox then
2297   attr[#attr+1] = format("/BBox[%s]", opts.bbox)
2298   grtype = 2
2299 end
2300 if opts.matrix then
2301   attr[#attr+1] = format("/Matrix[%s]", opts.matrix)
2302   grtype = opts.bbox and 4 or 1
2303 end
2304 if opts.asgroup then
2305   local t = { isolated = false, knockout = false }
2306   for _,v in ipairs(opts.asgroup:explode",+") do t[v] = true end
2307   attr[#attr+1] = format("/Group<</S/Transparency/I %s/K %s>>", t.isolated, t.knockout)
2308 end
2309 local trgroup = pdfetcs.tr_group
2310 trgroup.shifts[name] = { get_macro'MPlIx', get_macro'MPlly' }
2311 local whd
2312 local urx, ury = wd/factor, (ht+dp)/factor
2313 if pdfmode then
2314   attr = tableconcat(attr) :gsub(decimals,rmzeros)
2315   local index = tex.saveboxresource(boxid, attr, res, true, grtype)
2316   token.set_macro("luamplib.group"..name, tableconcat{
2317     "\\mplibstarttoPDF{0}{0}{",urx,"}{",ury,"}",
2318     "\\useboxresource ", index,
2319     "\\mplibstoptoPDF",
2320   }, "global")
2321   whd = format("%.3f %.3f 0", urx, ury) :gsub(decimals,rmzeros)
2322 else
2323   trgroup.cnt = (trgroup.cnt or 0) + 1
2324   local objname = format("@mplibtrgr%s", trgroup.cnt)
2325   texpstr {
2326     "\\expandafter\\newbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2327     "\\global\\setbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2328     "\\hbox{\\unhbox ", boxid, "}\\luamplibatnextshipout{",
2329     "\\special{pdf:bcontent}",
2330     "\\special{pdf:bxobj ", objname, " width ", wd, "sp height ", ht, "sp depth ", dp, "sp}",
2331     "\\unhbox\\csname luamplib.groupbox.", trgroup.cnt, "\\endcsname",
2332     "\\special{pdf:put @resources <<, res, ">>}",
2333     "\\special{pdf:exobj <<, tableconcat(attr), ">>}",
2334     "\\special{pdf:econtent}}",
2335   }
2336   token.set_macro("luamplib.group"..name, tableconcat{
2337     "\\mplibstarttoPDF{0}{0}{",urx,"}{",ury,"}",
2338     "\\setbox\\mplibscratchbox\\hbox{\\special{pdf:uxobj ", objname, "}}",
2339     "\\wd\\mplibscratchbox ", wd, "sp",
2340     "\\ht\\mplibscratchbox ", ht, "sp",
2341     "\\dp\\mplibscratchbox ", dp, "sp",
2342     "\\box\\mplibscratchbox",
2343     "\\mplibstoptoPDF",
2344   }, "global")

```

```

2345   whd = format("%.3f %.3f %.3f", urx, ht/factor, dp/factor) :gsub(decimals,rmzeros)
2346 end
2347 info("w/h/d of group '%s': %s", name, whd)
2348 end
2349
2350 local function stop_special_effects(fade,opaq,over)
2351   if fade then -- fading
2352     stop_pdf_code()
2353   end
2354   if opaq then -- opacity
2355     pdf_literalcode(opaq)
2356   end
2357   if over then -- color
2358     put2output"\special{pdf:ec}"
2359   end
2360 end
2361

```

Codes below for inserting PDF lieterals are mostly from ConTeXt general, with small changes when needed.

```

2362 local function getobjects(result,figure,f)
2363   return figure:objects()
2364 end
2365
2366 function luamplib.convert (result, flusher)
2367   luamplib.flush(result, flusher)
2368   return true -- done
2369 end
2370
2371 local function pdf_textfigure(font,size,text,width,height,depth)
2372   text = text:gsub(".",function(c)
2373     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost : false
2374   end)
2375   put2output("\mplibtexttext{%s}{%f}{%s}{%s}{%s}",font,size,text,0,0)
2376 end
2377
2378 local bend_tolerance = 131/65536
2379
2380 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
2381
2382 local function pen_characteristics(object)
2383   local t = mplib.pen_info(object)
2384   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
2385   divider = sx*sy - rx*ry
2386   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
2387 end
2388
2389 local function concat(px, py) -- no tx, ty here
2390   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
2391 end
2392
2393 local function curved(ith,pth)
2394   local d = pth.left_x - ith.right_x
2395   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance t

```

```

2396     d = pth.left_y - ith.right_y
2397     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance
2398         return false
2399     end
2400 end
2401 return true
2402 end
2403
2404 local function flushnormalpath(path,open)
2405     local pth, ith
2406     for i=1,#path do
2407         pth = path[i]
2408         if not ith then
2409             pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
2410         elseif curved(ith,pth) then
2411             pdf_literalcode("%f %f %f %f %f %f c",ith.right_x,ith.right_y,pth.left_x,pth.left_y,pth.x_coord,pth.y_coord)
2412         else
2413             pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
2414         end
2415         ith = pth
2416     end
2417     if not open then
2418         local one = path[1]
2419         if curved(pth,one) then
2420             pdf_literalcode("%f %f %f %f %f %f c",pth.right_x,pth.right_y,one.left_x,one.left_y,one.x_coord,one.y_coord)
2421         else
2422             pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2423         end
2424     elseif #path == 1 then -- special case .. draw point
2425         local one = path[1]
2426         pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
2427     end
2428 end
2429
2430 local function flushconcatpath(path,open)
2431     pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx, ty)
2432     local pth, ith
2433     for i=1,#path do
2434         pth = path[i]
2435         if not ith then
2436             pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
2437         elseif curved(ith,pth) then
2438             local a, b = concat(ith.right_x,ith.right_y)
2439             local c, d = concat(pth.left_x,pth.left_y)
2440             pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
2441         else
2442             pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
2443         end
2444         ith = pth
2445     end
2446     if not open then
2447         local one = path[1]
2448         if curved(pth,one) then
2449             local a, b = concat(pth.right_x,pth.right_y)

```

```

2450     local c, d = concat(one.left_x,one.left_y)
2451     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
2452     else
2453         pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2454     end
2455 elseif #path == 1 then -- special case .. draw point
2456     local one = path[1]
2457     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
2458 end
2459 end
2460

```

Finally, flush figures by inserting PDF literals.

```

2461 function luamplib.flush (result,flusher)
2462 if result then
2463     local figures = result.fig
2464     if figures then
2465         for f=1, #figures do
2466             info("flushing figure %s",f)
2467             local figure = figures[f]
2468             local objects = getobjects(result,figure,f)
2469             local fignum = tonumber(figure:filename():match("([%d]+)$") or figure:charcode() or 0)
2470             local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2471             local bbox = figure:boundingbox()
2472             local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
2473             if urx < llx then

```

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

```

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

2474     else

```

For legacy behavior, insert 'pre-fig' TeX code here.

```

2475         if tex_code_pre_mplib[f] then
2476             put2output(tex_code_pre_mplib[f])
2477         end
2478         pdf_startfigure(fignum,llx,lly,urx,ury)
2479         start_pdf_code()
2480         if objects then
2481             local savedpath = nil
2482             local savedhtap = nil
2483             for o=1,#objects do
2484                 local object      = objects[o]
2485                 local objecttype  = object.type

```

The following 9 lines are part of `btex...etex` patch. Again, colors are processed at this stage.

```

2486             local prescript      = object.prescript
2487             prescript = prescript and script2table(prescript) -- prescript is now a table
2488             local cr_over = do_preobj_CR(object,prescript) -- color
2489             local tr_opaq = do_preobj_TR(object,prescript) -- opacity
2490             local fading_ = do_preobj_FADE(object,prescript) -- fading

```



```

2491     local trgroup = do_preobj_GRP(object,prescript) -- transparency group
2492     local pattern_ = do_preobj_PAT(object,prescript) -- pattern
2493     if prescript and prescript.mplibtexboxid then
2494         put_tex_boxes(object,prescript)
2495     elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
2496     elseif objecttype == "start_clip" then
2497         local evenodd = not object.istext and object.postscript == "evenodd"
2498         start_pdf_code()
2499         flushnormalpath(object.path,false)
2500         pdf_literalcode(evenodd and "W* n" or "W n")
2501     elseif objecttype == "stop_clip" then
2502         stop_pdf_code()
2503         miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2504     elseif objecttype == "special" then

```

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

```

2505         if prescript and prescript.postmplibverbtx then
2506             figcontents.post[#figcontents.post+1] = prescript.postmplibverbtx
2507         end
2508     elseif objecttype == "text" then
2509         local ot = object.transform -- 3,4,5,6,1,2
2510         start_pdf_code()
2511         pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
2512         pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
2513         stop_pdf_code()
2514     elseif not trgroup and fading_ ~= "stop" then
2515         local evenodd, collect, both = false, false, false
2516         local postscript = object.postscript
2517         if not object.istext then
2518             if postscript == "evenodd" then
2519                 evenodd = true
2520             elseif postscript == "collect" then
2521                 collect = true
2522             elseif postscript == "both" then
2523                 both = true
2524             elseif postscript == "eoboth" then
2525                 evenodd = true
2526                 both = true
2527             end
2528         end
2529         if collect then
2530             if not savedpath then
2531                 savedpath = { object.path or false }
2532                 savedhtap = { object.htap or false }
2533             else
2534                 savedpath[#savedpath+1] = object.path or false
2535                 savedhtap[#savedhtap+1] = object.htap or false
2536             end
2537         else

```

Removed from ConTeXt general: color stuff.

```

2538         local ml = object.miterlimit
2539         if ml and ml ~= miterlimit then
2540             miterlimit = ml
2541             pdf_literalcode("%f M",ml)

```

```

2542     end
2543     local lj = object.linejoin
2544     if lj and lj ~= linejoin then
2545         linejoin = lj
2546         pdf_literalcode("%i j",lj)
2547     end
2548     local lc = object.linecap
2549     if lc and lc ~= linecap then
2550         linecap = lc
2551         pdf_literalcode("%i J",lc)
2552     end
2553     local dl = object.dash
2554     if dl then
2555         local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
2556         if d ~= dashed then
2557             dashed = d
2558             pdf_literalcode(dashed)
2559         end
2560     elseif dashed then
2561         pdf_literalcode("[] 0 d")
2562         dashed = false
2563     end
2564     local path = object.path
2565     local transformed, penwidth = false, 1
2566     local open = path and path[1].left_type and path[#path].right_type
2567     local pen = object.pen
2568     if pen then
2569         if pen.type == 'elliptical' then
2570             transformed, penwidth = pen_characteristics(object) -- boolean, value
2571             pdf_literalcode("%f w",penwidth)
2572             if objecttype == 'fill' then
2573                 objecttype = 'both'
2574             end
2575         else -- calculated by mplib itself
2576             objecttype = 'fill'
2577         end
2578     end

```

Added : shading

```

2579     local shade_no = do_preobj_SH(object,prescript) -- shading
2580     if shade_no then
2581         pdf_literalcode"q /Pattern cs"
2582         objecttype = false
2583     end
2584     if transformed then
2585         start_pdf_code()
2586     end
2587     if path then
2588         if savedpath then
2589             for i=1,#savedpath do
2590                 local path = savedpath[i]
2591                 if transformed then
2592                     flushconcatpath(path,open)
2593                 else
2594                     flushnormalpath(path,open)

```

```

2595         end
2596     end
2597     savedpath = nil
2598 end
2599 if transformed then
2600     flushconcatpath(path,open)
2601 else
2602     flushnormalpath(path,open)
2603 end
2604 if objecttype == "fill" then
2605     pdf_literalcode(evenodd and "h f*" or "h f")
2606 elseif objecttype == "outline" then
2607     if both then
2608         pdf_literalcode(evenodd and "h B*" or "h B")
2609     else
2610         pdf_literalcode(open and "S" or "h S")
2611     end
2612 elseif objecttype == "both" then
2613     pdf_literalcode(evenodd and "h B*" or "h B")
2614 end
2615 end
2616 if transformed then
2617     stop_pdf_code()
2618 end
2619 local path = object.htap

```

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

```

2620 if path then
2621     if transformed then
2622         start_pdf_code()
2623     end
2624     if savedhtap then
2625         for i=1,#savedhtap do
2626             local path = savedhtap[i]
2627             if transformed then
2628                 flushconcatpath(path,open)
2629             else
2630                 flushnormalpath(path,open)
2631             end
2632         end
2633         savedhtap = nil
2634         evenodd = true
2635     end
2636     if transformed then
2637         flushconcatpath(path,open)
2638     else
2639         flushnormalpath(path,open)
2640     end
2641     if objecttype == "fill" then
2642         pdf_literalcode(evenodd and "h f*" or "h f")
2643     elseif objecttype == "outline" then
2644         pdf_literalcode(open and "S" or "h S")
2645     elseif objecttype == "both" then
2646         pdf_literalcode(evenodd and "h B*" or "h B")
2647     end

```

```

2648         if transformed then
2649             stop_pdf_code()
2650         end
2651     end

```

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

```

2652         if shade_no then -- shading
2653             pdf_literalcode("W% s n /MPLibSh% s sh Q",evenodd and "*" or "",shade_no)
2654         end
2655     end
2656 end
2657 if fading_ == "start" then
2658     pdfetcs.fading.specialeffects = {fading_, tr_opaq, cr_over}
2659 elseif trgroup == "start" then
2660     pdfetcs.tr_group.specialeffects = {fading_, tr_opaq, cr_over}
2661 elseif fading_ == "stop" then
2662     local se = pdfetcs.fading.specialeffects
2663     if se then stop_special_effects(se[1], se[2], se[3]) end
2664 elseif trgroup == "stop" then
2665     local se = pdfetcs.tr_group.specialeffects
2666     if se then stop_special_effects(se[1], se[2], se[3]) end
2667 else
2668     stop_special_effects(fading_, tr_opaq, cr_over)
2669 end
2670 if fading_ or trgroup then -- extgs resetted
2671     miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
2672 end
2673 end
2674 end
2675 stop_pdf_code()
2676 pdf_stopfigure()

```

output collected materials to PDF, plus legacy verbatimex code.

```

2677     for _,v in ipairs(figcontents) do
2678         if type(v) == "table" then
2679             texsprint"\mplibtoPDF{"; texsprint(v[1], v[2]); texsprint"}"
2680         else
2681             texsprint(v)
2682         end
2683     end
2684     if #figcontents.post > 0 then texsprint(figcontents.post) end
2685     figcontents = { post = { } }
2686 end
2687 end
2688 end
2689 end
2690 end
2691
2692 function luamplib.colorconverter (cr)
2693     local n = #cr
2694     if n == 4 then
2695         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
2696         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
2697     elseif n == 3 then

```

```

2698   local r, g, b = cr[1], cr[2], cr[3]
2699   return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
2700 else
2701   local s = cr[1]
2702   return format("%.3f g %.3f G",s,s), "0 g 0 G"
2703 end
2704 end

```

2.2 T_EX package

First we need to load some packages.

```
2705 \ifcsname ProvidesPackage\endcsname
```

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

```

2706 \NeedsTeXFormat{LaTeX2e}
2707 \ProvidesPackage{luamplib}
2708   [2024/11/25 v2.35.1 mplib package for LuaTeX]
2709 \fi
2710 \ifdefined\newluafunction\else
2711   \input ltuatex
2712 \fi

```

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

```

2713 \ifnum\outputmode=0
2714   \ifdefined\AddToHookNext
2715     \def\luamplibatnextshipout{\AddToHookNext{shipout/background}}
2716     \def\luamplibatfirstshipout{\AddToHook{shipout/firstpage}}
2717     \def\luamplibateveryshipout{\AddToHook{shipout/background}}
2718   \else
2719     \input atbegshi.sty
2720     \def\luamplibatnextshipout#1{\AtBeginShipoutNext{\AtBeginShipoutAddToBox{#1}}}
2721     \let\luamplibatfirstshipout\AtBeginShipoutFirst
2722     \def\luamplibateveryshipout#1{\AtBeginShipout{\AtBeginShipoutAddToBox{#1}}}
2723   \fi
2724 \fi

```

Loading of lua code.

```
2725 \directlua{require("luamplib")}
```

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

```

2726 \ifx\pdfoutput\undefined
2727   \let\pdfoutput\outputmode
2728 \fi
2729 \ifx\pdfliteral\undefined
2730   \protected\def\pdfliteral{\pdfextension literal}
2731 \fi

```

Set the format for METAPOST.

```
2732 \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.

```

2733 \ifnum\pdfoutput>0
2734 \let\mplibtoPDF\pdfliteral
2735 \else
2736 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
2737 \ifcsname PackageInfo\endcsname
2738 \PackageInfo{luamplib}{only dvipdfmx is supported currently}
2739 \else
2740 \immediate\write-1{luamplib Info: only dvipdfmx is supported currently}
2741 \fi
2742 \fi

```

To make mplibcode typeset always in horizontal mode.

```

2743 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
2744 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
2745 \mplibnoforcehmode

```

Catcode. We want to allow comment sign in mplibcode.

```

2746 \def\mplibsetupcatcodes{%
2747 %catcode`\{=12 %catcode`\}=12
2748 \catcode`\#=12 \catcode`\^=12 \catcode`\~=12 \catcode`\_=12
2749 \catcode`\&=12 \catcode`\$=12 \catcode`\%=12 \catcode`\^M=12
2750 }

```

Make btex...etex box zero-metric.

```

2751 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}

```

use Transparency Group

```

2752 \protected\def\usemplibgroup#1#1{\usemplibgroupmain}
2753 \def\usemplibgroupmain#1{\csname luamplib.group.#1\endcsname}
2754 \protected\def\mplibgroup#1{%
2755 \begingroup
2756 \def\MPllx{0}\def\MPlly{0}%
2757 \def\mplibgroupname{#1}%
2758 \mplibgroupgetnexttok
2759 }
2760 \def\mplibgroupgetnexttok{\futurelet\nexttok\mplibgroupbranch}
2761 \def\mplibgroupskipspace{\afterassignment\mplibgroupgetnexttok\let\nexttok=}
2762 \def\mplibgroupbranch{%
2763 \ifx [\nexttok
2764 \expandafter\mplibgroupopts
2765 \else
2766 \ifx\mplibsptoken\nexttok
2767 \expandafter\expandafter\expandafter\mplibgroupskipspace
2768 \else
2769 \let\mplibgroupoptions\empty
2770 \expandafter\expandafter\expandafter\mplibgroupmain
2771 \fi
2772 \fi
2773 }
2774 \def\mplibgroupopts[#1]{\def\mplibgroupoptions{#1}\mplibgroupmain}
2775 \def\mplibgroupmain{\setbox\mplibscratchbox\hbox\bgroup\ignorespaces}
2776 \protected\def\endmplibgroup{\egroup
2777 \directlua{ luamplib.registergroup(
2778 \the\mplibscratchbox, '\mplibgroupname', {\mplibgroupoptions}
2779 )}%
2780 \endgroup

```

```

2781 }
      Patterns
2782 {\def\:\global\let\mplibsptoken= }\: }
2783 \protected\def\mppattern#1{%
2784 \begingroup
2785 \def\mplibpatternname{#1}%
2786 \mplibpatterngetnexttok
2787 }
2788 \def\mplibpatterngetnexttok{\futurelet\nexttok\mplibpatternbranch}
2789 \def\mplibpatternskipsspace{\afterassignment\mplibpatterngetnexttok\let\nexttok= }
2790 \def\mplibpatternbranch{%
2791 \ifx [\nexttok
2792 \expandafter\mplibpatternopts
2793 \else
2794 \ifx\mplibsptoken\nexttok
2795 \expandafter\expandafter\expandafter\mplibpatternskipsspace
2796 \else
2797 \let\mplibpatternoptions\empty
2798 \expandafter\expandafter\expandafter\mplibpatternmain
2799 \fi
2800 \fi
2801 }
2802 \def\mplibpatternopts[#1]{%
2803 \def\mplibpatternoptions{#1}%
2804 \mplibpatternmain
2805 }
2806 \def\mplibpatternmain{%
2807 \setbox\mplibscratchbox\hbox\bgroup\ignorespaces
2808 }
2809 \protected\def\endmppattern{%
2810 \egroup
2811 \directlua{ luamplib.registerpattern(
2812 \the\mplibscratchbox, '\mplibpatternname', {\mplibpatternoptions}
2813 )}%
2814 \endgroup
2815 }

      simple way to use mplib: \mpfig draw fullcircle scaled 10; \endmpfig
2816 \def\mpfiginstancename{@mpfig}
2817 \protected\def\mpfig{%
2818 \begingroup
2819 \futurelet\nexttok\mplibmpfigbranch
2820 }
2821 \def\mplibmpfigbranch{%
2822 \ifx *\nexttok
2823 \expandafter\mplibprempfig
2824 \else
2825 \ifx [\nexttok
2826 \expandafter\expandafter\expandafter\mplibgobbleoptsmfig
2827 \else
2828 \expandafter\expandafter\expandafter\mplibmainmpfig
2829 \fi
2830 \fi
2831 }

```

```

2832 \def\mplibobbleoptsmpfig[#1]{\mplibmainmpfig}
2833 \def\mplibmainmpfig{%
2834   \begingroup
2835   \mplibsetupcatcodes
2836   \mplibdomainmpfig
2837 }
2838 \long\def\mplibdomainmpfig#1\endmpfig{%
2839   \endgroup
2840   \directlua{
2841     local legacy = luamplib.legacyverbatim
2842     local everympfig = luamplib.everymplib["\mpfiginstancename"] or ""
2843     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"] or ""
2844     luamplib.legacyverbatim = false
2845     luamplib.everymplib["\mpfiginstancename"] = ""
2846     luamplib.everyendmplib["\mpfiginstancename"] = ""
2847     luamplib.process_mplibcode(
2848       "beginfig(0) ".everympfig.." ".[===[\unexpanded{#1}]===].." ".everyendmpfig.." endfig;",
2849       "\mpfiginstancename")
2850     luamplib.legacyverbatim = legacy
2851     luamplib.everymplib["\mpfiginstancename"] = everympfig
2852     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
2853   }%
2854   \endgroup
2855 }
2856 \def\mplibprempfig#1{%
2857   \begingroup
2858   \mplibsetupcatcodes
2859   \mplibdoprempfig
2860 }
2861 \long\def\mplibdoprempfig#1\endmpfig{%
2862   \endgroup
2863   \directlua{
2864     local legacy = luamplib.legacyverbatim
2865     local everympfig = luamplib.everymplib["\mpfiginstancename"]
2866     local everyendmpfig = luamplib.everyendmplib["\mpfiginstancename"]
2867     luamplib.legacyverbatim = false
2868     luamplib.everymplib["\mpfiginstancename"] = ""
2869     luamplib.everyendmplib["\mpfiginstancename"] = ""
2870     luamplib.process_mplibcode([===[\unexpanded{#1}]===], "\mpfiginstancename")
2871     luamplib.legacyverbatim = legacy
2872     luamplib.everymplib["\mpfiginstancename"] = everympfig
2873     luamplib.everyendmplib["\mpfiginstancename"] = everyendmpfig
2874   }%
2875   \endgroup
2876 }
2877 \protected\def\endmpfig{endmpfig}

```

The Plain-specific stuff.

```

2878 \unless\ifcsname ver@luamplib.sty\endcsname
2879   \def\mplibcodegetinstancename[#1]{\gdef\currentmpinstancename{#1}\mplibcodeindeed}
2880   \protected\def\mplibcode{%
2881     \begingroup
2882     \futurelet\nexttok\mplibcodebranch
2883   }
2884   \def\mplibcodebranch{%

```



```

2885 \ifx [\nexttok
2886 \expandafter\mplibcodegetinstancename
2887 \else
2888 \global\let\currentmpinstancename\empty
2889 \expandafter\mplibcodeindeed
2890 \fi
2891 }
2892 \def\mplibcodeindeed{%
2893 \begingroup
2894 \mplibsetupcatcodes
2895 \mplibdocode
2896 }
2897 \long\def\mplibdocode#1\endmplibcode{%
2898 \endgroup
2899 \directlua{luamplib.process_mplibcode([==[\unexpanded{#1}]===],"\currentmpinstancename")}%
2900 \endgroup
2901 }
2902 \protected\def\endmplibcode{endmplibcode}
2903 \else
    The  $\TeX$ -specific part: a new environment.
2904 \newenvironment{mplibcode}[1][{%
2905 \global\def\currentmpinstancename{#1}%
2906 \mplibtmptoks}\ltxdomplibcode
2907 }{}
2908 \def\ltxdomplibcode{%
2909 \begingroup
2910 \mplibsetupcatcodes
2911 \ltxdomplibcodeindeed
2912 }
2913 \def\mplib@mplibcode{mplibcode}
2914 \long\def\ltxdomplibcodeindeed#1\end#2{%
2915 \endgroup
2916 \mplibtmptoks\expandafter{\the\mplibtmptoks#1}%
2917 \def\mplibtemp@a{#2}%
2918 \ifx\mplib@mplibcode\mplibtemp@a
2919 \directlua{luamplib.process_mplibcode([==[\the\mplibtmptoks]===],"\currentmpinstancename")}%
2920 \end{mplibcode}%
2921 \else
2922 \mplibtmptoks\expandafter{\the\mplibtmptoks\end{#2}}%
2923 \expandafter\ltxdomplibcode
2924 \fi
2925 }
2926 \fi

    User settings.
2927 \def\mplibshowlog#1{\directlua{
2928 local s = string.lower("#1")
2929 if s == "enable" or s == "true" or s == "yes" then
2930 luamplib.showlog = true
2931 else
2932 luamplib.showlog = false
2933 end
2934 }}
2935 \def\mpliblegacybehavior#1{\directlua{

```

```

2936 local s = string.lower("#1")
2937 if s == "enable" or s == "true" or s == "yes" then
2938     luamplib.legacyverbatimex = true
2939 else
2940     luamplib.legacyverbatimex = false
2941 end
2942 }}
2943 \def\mplibverbatim#1{\directlua{
2944     local s = string.lower("#1")
2945     if s == "enable" or s == "true" or s == "yes" then
2946         luamplib.verbatiminput = true
2947     else
2948         luamplib.verbatiminput = false
2949     end
2950 }}
2951 \newtoks\mplibmptoks
    \everymplib & \everyendmplib: macros resetting luamplib.every(end)mplib tables
2952 \ifcsname ver@luamplib.sty\endcsname
2953 \protected\def\everymplib{%
2954     \begingroup
2955     \mplibsetupcatcodes
2956     \mplibdoeverymplib
2957 }
2958 \protected\def\everyendmplib{%
2959     \begingroup
2960     \mplibsetupcatcodes
2961     \mplibdoeveryendmplib
2962 }
2963 \newcommand\mplibdoeverymplib[2][]{%
2964     \endgroup
2965     \directlua{
2966         luamplib.everymplib["#1"] = [===[\unexpanded{#2}]===]
2967     }%
2968 }
2969 \newcommand\mplibdoeveryendmplib[2][]{%
2970     \endgroup
2971     \directlua{
2972         luamplib.everyendmplib["#1"] = [===[\unexpanded{#2}]===]
2973     }%
2974 }
2975 \else
2976 \def\mplibgetinstancename[#1]{\def\currentmpinstancename{#1}}
2977 \protected\def\everymplib#1{%
2978     \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
2979     \begingroup
2980     \mplibsetupcatcodes
2981     \mplibdoeverymplib
2982 }
2983 \long\def\mplibdoeverymplib#1{%
2984     \endgroup
2985     \directlua{
2986         luamplib.everymplib["\currentmpinstancename"] = [===[\unexpanded{#1}]===]
2987     }%

```

```

2988 }
2989 \protected\def\everyendmplib#1#1{%
2990 \ifx\empty#1\empty \mplibgetinstancename[]\else \mplibgetinstancename#1\fi
2991 \begingroup
2992 \mplibsetupcatcodes
2993 \mplibdoeveryendmplib
2994 }
2995 \long\def\mplibdoeveryendmplib#1#1{%
2996 \endgroup
2997 \directlua{
2998   luamplib.everyendmplib["\currentmpinstancename"] = [====[\unexpanded{#1}]====]
2999 }%
3000 }
3001 \fi

```

Allow \TeX `dimen/color` macros. Now `runscript` does the job, so the following lines are not needed for most cases.

```

3002 \def\mpdim#1{ runscript("luamplibdimen{#1}") }
3003 \def\mpcolor#1#1{\domplibcolor{#1}}
3004 \def\domplibcolor#1#2{ runscript("luamplibcolor{#1{#2}}") }

```

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

```

3005 \def\mplibnumbersystem#1{\directlua{
3006   local t = "#1"
3007   if t == "binary" then t = "decimal" end
3008   luamplib.numbersystem = t
3009 }}

```

Settings for `.mp` cache files.

```

3010 \def\mplibmakenocache#1{\mplibdomakenocache #1,*}
3011 \def\mplibdomakenocache#1,{%
3012 \ifx\empty#1\empty
3013 \expandafter\mplibdomakenocache
3014 \else
3015 \ifx*#1\else
3016 \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
3017 \expandafter\expandafter\expandafter\mplibdomakenocache
3018 \fi
3019 \fi
3020 }
3021 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*}
3022 \def\mplibdocancelnocache#1,{%
3023 \ifx\empty#1\empty
3024 \expandafter\mplibdocancelnocache
3025 \else
3026 \ifx*#1\else
3027 \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
3028 \expandafter\expandafter\expandafter\mplibdocancelnocache
3029 \fi
3030 \fi
3031 }
3032 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

3033 \def\mplibtexttextlabel#1{\directlua{

```

```

3034 local s = string.lower("#1")
3035 if s == "enable" or s == "true" or s == "yes" then
3036   luamplib.texttextlabel = true
3037 else
3038   luamplib.texttextlabel = false
3039 end
3040 }}
3041 \def\mplibcodeinherit#1{\directlua{
3042   local s = string.lower("#1")
3043   if s == "enable" or s == "true" or s == "yes" then
3044     luamplib.codeinherit = true
3045   else
3046     luamplib.codeinherit = false
3047   end
3048 }}
3049 \def\mplibglobaltexttext#1{\directlua{
3050   local s = string.lower("#1")
3051   if s == "enable" or s == "true" or s == "yes" then
3052     luamplib.globaltexttext = true
3053   else
3054     luamplib.globaltexttext = false
3055   end
3056 }}

```

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

```

3057 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

3058 \def\mplibstarttoPDF#1#2#3#4{%
3059   \prependtomplibbox
3060   \hbox dir TLT\bgroup
3061   \xdef\MPllx{#1}\xdef\MPlly{#2}%
3062   \xdef\MPurx{#3}\xdef\MPury{#4}%
3063   \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3064   \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3065   \parskip0pt%
3066   \leftskip0pt%
3067   \parindent0pt%
3068   \everypar{}%
3069   \setbox\mplibscratchbox\vbox\bgroup
3070   \noindent
3071 }
3072 \def\mplibstoptoPDF{%
3073   \par
3074   \egroup %
3075   \setbox\mplibscratchbox\hbox %
3076   {\hskip-\MPllx bp%
3077    \raise-\MPlly bp%
3078    \box\mplibscratchbox}%
3079   \setbox\mplibscratchbox\vbox to \MPheight
3080   {\vfill
3081    \hsize\MPwidth
3082    \wd\mplibscratchbox0pt%
3083    \ht\mplibscratchbox0pt%

```

```

3084 \dp\mplibscratchbox0pt%
3085 \box\mplibscratchbox}%
3086 \wd\mplibscratchbox\MPwidth
3087 \ht\mplibscratchbox\MPheight
3088 \box\mplibscratchbox
3089 \egroup
3090 }

```

Text items have a special handler.

```

3091 \def\mplibtexttext#1#2#3#4#5{%
3092 \begingroup
3093 \setbox\mplibscratchbox\hbox
3094 {\font\temp=#1 at #2bp%
3095 \temp
3096 #3}%
3097 \setbox\mplibscratchbox\hbox
3098 {\hskip#4 bp%
3099 \raise#5 bp%
3100 \box\mplibscratchbox}%
3101 \wd\mplibscratchbox0pt%
3102 \ht\mplibscratchbox0pt%
3103 \dp\mplibscratchbox0pt%
3104 \box\mplibscratchbox
3105 \endgroup
3106 }

```

Input luamplib.cfg when it exists.

```

3107 \openin0=luamplib.cfg
3108 \ifeof0 \else
3109 \closein0
3110 \input luamplib.cfg
3111 \fi

```

Code for tagpdf

```

3112 \def\luamplibtagtextbegin#1{}
3113 \let\luamplibtagtextend\relax
3114 \ifcsname SuspendTagging\endcsname\else\endinput\fi
3115 \ifcsname ver@tagpdf.sty\endcsname \else
3116 \ExplSyntaxOn
3117 \keys_define:nn{luamplib/notag}
3118 {
3119 ,alt .code:n = { }
3120 ,actualtext .code:n = { }
3121 ,artifact .code:n = { }
3122 ,text .code:n = { }
3123 ,correct-BBox .code:n = { }
3124 ,tag .code:n = { }
3125 ,debug .code:n = { }
3126 ,instance .code:n = { \tl_gset:Nn \currentmpinstancename {#1} }
3127 ,instancename .meta:n = { instance = {#1} }
3128 ,unknown .code:n = { \tl_gset:Ne \currentmpinstancename {\l_keys_key_str} }
3129 }
3130 \RenewDocumentCommand\mplibcode{0{}}
3131 {
3132 \tl_gset_eq:NN \currentmpinstancename \c_empty_tl

```

```

3133     \keys_set:nn{luamplib/notag}{#1}
3134     \mplibmptoks{}\ltxdomplibcode
3135   }
3136   \ExplSyntaxOff
3137   \let\mplibaltext \luamplibtagtextbegin
3138   \let\mplibactualtext \mplibaltext
3139   \endinput\fi
3140 \let\mplibstarttoPDForiginal\mplibstarttoPDF
3141 \let\mplibstoptoPDForiginal\mplibstoptoPDF
3142 \let\mplibputtextboxoriginal\mplibputtextbox
3143 \ExplSyntaxOn
3144 \tl_new:N \l__luamplib_tag_alt_tl
3145 \tl_new:N \l__luamplib_tag_alt_dflt_tl
3146 \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure}
3147 \tl_new:N \l__luamplib_tag_actual_tl
3148 \tl_new:N \l__luamplib_tag_struct_tl
3149 \tl_set:Nn\l__luamplib_tag_struct_tl {Figure}
3150 \bool_new:N \l__luamplib_tag_usertext_bool
3151 \bool_new:N \l__luamplib_tag_BBox_bool
3152 \bool_set_true:N \l__luamplib_tag_BBox_bool
3153 \seq_new:N\l__luamplib_tag_bboxcorr_seq
3154 \bool_new:N\l__luamplib_tag_bboxcorr_bool
3155 \bool_new:N \l__luamplib_tag_debug_bool
3156 \tl_new:N \l__luamplib_BBox_label_tl
3157 \tl_new:N \l__luamplib_BBox_llx_tl
3158 \tl_new:N \l__luamplib_BBox_lly_tl
3159 \tl_new:N \l__luamplib_BBox_urx_tl
3160 \tl_new:N \l__luamplib_BBox_ury_tl
3161 \cs_set_nopar:Npn \luamplibtagtextbegin #1
3162 {
3163   \bool_if:NTF \l__luamplib_tag_usertext_bool
3164   {
3165     \tag_mc_end_push:
3166     \tag_mc_begin:n{}
3167     \tag_struct_begin:n{tag=NonStruct,stash}
3168     \def\myboxnum{#1}
3169     \edef\mystructnum{\tag_get:n{struct_num}}
3170     \edef\statebeforebox{\inteval{\tag_get:n{struct_counter}+\tag_get:n{mc_counter}}}
3171   }
3172   {
3173     \tag_if_active:TF
3174     { \chardef\mplibtmpnum\@ne }
3175     { \chardef\mplibtmpnum\z@ }
3176     \SuspendTagging{luamplib.texttext}
3177   }
3178 }
3179 \cs_set_nopar:Npn \luamplibtagtextend
3180 {
3181   \bool_if:NTF \l__luamplib_tag_usertext_bool
3182   {
3183     \edef\stateafterbox{\inteval{\tag_get:n{struct_counter}+\tag_get:n{mc_counter}}}
3184     \tag_if_active:T {
3185       \int_compare:nNnTF
3186       {\stateafterbox}

```

```

3187     =
3188     {\statebeforebox}
3189     { \cs_gset_nopar:cpe {luamplib.notagbox.\myboxnum} {\mystructnum} }
3190     { \cs_gset_nopar:cpe {luamplib.tagbox.\myboxnum} {\mystructnum} }
3191   }
3192   \tag_struct_end:
3193   \tag_mc_end:
3194   \tag_mc_begin_pop:n{}
3195 }
3196 {
3197   \ifnum\mplibtmpnum=\@ne
3198     \ResumeTagging{luamplib.texttext}
3199   \fi
3200 }
3201 }
3202 \msg_new:nnn {luamplib}{figure-text-reuse}
3203 {
3204   texttext~box~#1~probably~is~incorrectly~tagged.\\
3205   Reusing~a~box~in~text~keyed~figures~is~strongly~discouraged.
3206 }
3207 \cs_set_nopar:Npn \mplibputtextbox #1
3208 {
3209   \vbox to 0pt{\vss\hbox to 0pt{%
3210     \bool_if:NTF \l__luamplib_tag_usetext_bool
3211     {
3212       \ResumeTagging{luamplib.puttextbox}
3213       \tag_mc_end:
3214       \cs_if_exist:cTF {luamplib.tagbox.#1}
3215       {
3216         \tag_struct_use_num:n {\csname luamplib.tagbox.#1\endcsname}
3217         \raise\dp#1\copy#1
3218       }
3219       {
3220         \cs_if_exist:cF {luamplib.notagbox.#1}
3221         {
3222           \msg_warning:nnn{luamplib}{figure-text-reuse}{#1}
3223         }
3224         \tag_mc_begin:n{}
3225         \chardef\mplibtmpnum=#1\relax
3226         \tag_mc_reset_box:N \mplibtmpnum
3227         \raise\dp#1\copy#1
3228         \tag_mc_end:
3229       }
3230       \tag_mc_begin:n{artifact}
3231     }
3232     {
3233       \chardef\mplibtmpnum=#1\relax
3234       \tag_mc_reset_box:N \mplibtmpnum
3235       \raise\dp#1\copy#1
3236     }
3237   \hss}}
3238 }
3239 \cs_new_nopar:Npn \__luamplib_tagging_begin_figure:
3240 {

```

```

3241 \tag_if_active:T
3242 {
3243   \tag_mc_end_push:
3244   \tl_if_empty:NT\l__luamplib_tag_alt_tl
3245   {
3246     \msg_warning:nne{luamplib}{alt-text-missing}{\l__luamplib_tag_alt_dflt_tl}
3247     \tl_set:N\l__luamplib_tag_alt_tl {\l__luamplib_tag_alt_dflt_tl}
3248   }
3249   \tag_struct_begin:n
3250   {
3251     tag=\l__luamplib_tag_struct_tl,
3252     alt=\l__luamplib_tag_alt_tl,
3253   }
3254   \tag_mc_begin:n{}
3255 }
3256 }
3257 \cs_new_nopar:Npn \__luamplib_tagging_end_figure:
3258 {
3259   \tag_if_active:T
3260   {
3261     \tag_mc_end:
3262     \tag_struct_end:
3263     \tag_mc_begin_pop:n{}
3264   }
3265 }
3266 \cs_new_nopar:Npn \__luamplib_tagging_begin_actualtext:
3267 {
3268   \tag_if_active:T
3269   {
3270     \tag_mc_end_push:
3271     \tag_struct_begin:n
3272     {
3273       tag=Span,
3274       actualtext=\l__luamplib_tag_actual_tl,
3275     }
3276     \tag_mc_begin:n{}
3277   }
3278 }
3279 \cs_set_eq:NN \__luamplib_tagging_end_actualtext: \__luamplib_tagging_end_figure:
3280 \cs_new_nopar:Npn \__luamplib_tagging_begin_artifact:
3281 {
3282   \tag_if_active:T
3283   {
3284     \tag_mc_end_push:
3285     \tag_mc_begin:n{artifact}
3286   }
3287 }
3288 \cs_new_nopar:Npn \__luamplib_tagging_end_artifact:
3289 {
3290   \tag_if_active:T
3291   {
3292     \tag_mc_end:
3293     \tag_mc_begin_pop:n{}
3294   }

```



```

3295 }
3296 \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_figure:
3297 \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_figure:
3298 \keys_define:nn{luamplib/tag}
3299 {
3300   ,alt .code:n =
3301   {
3302     \bool_set_true:N \l__luamplib_tag_BBox_bool
3303     \bool_set_false:N \l__luamplib_tag_usetext_bool
3304     \tl_set:N\l__luamplib_tag_alt_tl{\text_purify:n{#1}}
3305   }
3306   ,actualtext .code:n =
3307   {
3308     \bool_set_false:N \l__luamplib_tag_BBox_bool
3309     \bool_set_false:N \l__luamplib_tag_usetext_bool
3310     \tl_set:N\l__luamplib_tag_actual_tl{\text_purify:n{#1}}
3311     \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_actualtext:
3312     \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_actualtext:
3313     \tag_if_active:T {\noindent}
3314   }
3315   ,artifact .code:n =
3316   {
3317     \bool_set_false:N \l__luamplib_tag_BBox_bool
3318     \bool_set_false:N \l__luamplib_tag_usetext_bool
3319     \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3320     \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3321   }
3322   ,text .code:n =
3323   {
3324     \bool_set_false:N \l__luamplib_tag_BBox_bool
3325     \bool_set_true:N \l__luamplib_tag_usetext_bool
3326     \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3327     \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3328     \tag_if_active:T {\noindent}
3329   }
3330   ,tag .code:n =
3331   {
3332     \str_case:nnF {#1}
3333     {
3334       {artifact}
3335       {
3336         \bool_set_false:N \l__luamplib_tag_BBox_bool
3337         \bool_set_false:N \l__luamplib_tag_usetext_bool
3338         \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3339         \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3340       }
3341       {text}
3342       {
3343         \bool_set_false:N \l__luamplib_tag_BBox_bool
3344         \bool_set_true:N \l__luamplib_tag_usetext_bool
3345         \cs_set_eq:NN \luamplibtaggingbegin \__luamplib_tagging_begin_artifact:
3346         \cs_set_eq:NN \luamplibtaggingend \__luamplib_tagging_end_artifact:
3347         \tag_if_active:T {\noindent}
3348       }
3349     }
3350   }

```

```

3349         {false}
3350         {
3351         \SuspendTagging{luamplib.tagfalse}
3352         }
3353     }
3354     {
3355     \tl_set:Nn\l__luamplib_tag_struct_tl{#1}
3356     }
3357 }
3358 ,correct-BBox .code:n =
3359 {
3360     \bool_set_true:N \l__luamplib_tag_bboxcorr_bool
3361     \seq_set_split:Nnn \l__luamplib_tag_bboxcorr_seq{~}{#1~0pt~0pt~0pt}
3362 }
3363 ,debug .code:n =
3364 { \bool_set_true:N \l__luamplib_tag_debug_bool }
3365 ,instance .code:n =
3366 { \tl_gset:Nn \currentmpinstancename {#1} }
3367 ,instancename .meta:n = { instance = {#1} }
3368 ,unknown .code:n =
3369 { \tl_gset:Ne \currentmpinstancename {\l_keys_key_str} }
3370 }
3371 \cs_new_nopar:Npn \luamplibtaggingBBox
3372 {
3373     \bool_lazy_and:nnT
3374     {\tag_if_active_p:}
3375     {\l__luamplib_tag_BBox_bool}
3376     {
3377     \tl_set:Ne \l__luamplib_BBox_label_tl {luamplib.BBox.\tag_get:n{struct_num}}
3378     \tex_savepos:D
3379     \property_record:ee{\l__luamplib_BBox_label_tl}{xpos,ypos,abspage}
3380     \tl_set:Ne \l__luamplib_BBox_llx_tl
3381     {
3382     \dim_to_decimal_in_bp:n
3383     { \property_ref:een {\l__luamplib_BBox_label_tl}{xpos}{0}sp }
3384     }
3385     \tl_set:Ne \l__luamplib_BBox_lly_tl
3386     {
3387     \dim_to_decimal_in_bp:n
3388     { \property_ref:een {\l__luamplib_BBox_label_tl}{ypos}{0}sp - \dp\mplibscratchbox }
3389     }
3390     \tl_set:Ne \l__luamplib_BBox_urx_tl
3391     {
3392     \dim_to_decimal_in_bp:n
3393     { \l__luamplib_BBox_llx_tl bp + \wd\mplibscratchbox }
3394     }
3395     \tl_set:Ne \l__luamplib_BBox_ury_tl
3396     {
3397     \dim_to_decimal_in_bp:n
3398     { \l__luamplib_BBox_lly_tl bp + \ht\mplibscratchbox + \dp\mplibscratchbox }
3399     }
3400     \bool_if:NT \l__luamplib_tag_bboxcorr_bool
3401     {
3402     \tl_set:Ne \l__luamplib_BBox_llx_tl

```

```

3403     {
3404         \fp_eval:n
3405         {
3406             \l__luamplib_BBox_llx_tl
3407             +
3408             \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {1} }
3409         }
3410     }
3411 \tl_set:Nc \l__luamplib_BBox_lly_tl
3412 {
3413     \fp_eval:n
3414     {
3415         \l__luamplib_BBox_lly_tl
3416         +
3417         \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {2} }
3418     }
3419 }
3420 \tl_set:Nc \l__luamplib_BBox_urx_tl
3421 {
3422     \fp_eval:n
3423     {
3424         \l__luamplib_BBox_urx_tl
3425         +
3426         \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {3} }
3427     }
3428 }
3429 \tl_set:Nc \l__luamplib_BBox_ury_tl
3430 {
3431     \fp_eval:n
3432     {
3433         \l__luamplib_BBox_ury_tl
3434         +
3435         \dim_to_decimal_in_bp:n {\seq_item:Nn \l__luamplib_tag_bboxcorr_seq {4} }
3436     }
3437 }
3438 }
3439 \prop_gput:cne
3440 { g__tag_struct_\tag_get:n{struct_num}_prop }
3441 {A}
3442 {
3443     << /0 /Layout /BBox [
3444         \l__luamplib_BBox_llx_tl\c_space_tl
3445         \l__luamplib_BBox_lly_tl\c_space_tl
3446         \l__luamplib_BBox_urx_tl\c_space_tl
3447         \l__luamplib_BBox_ury_tl
3448     ] >>
3449 }
3450 \bool_if:NT \l__luamplib_tag_debug_bool
3451 {
3452     \iow_log:e
3453     {
3454         luamplib/tag/debug:~BBox~of~structure~\tag_get:n{struct_num}~is~
3455         \l__luamplib_BBox_llx_tl\c_space_tl
3456         \l__luamplib_BBox_lly_tl\c_space_tl

```

```

3457         \l__luamplib_BBox_urx_tl\c_space_tl
3458         \l__luamplib_BBox_ury_tl
3459     }
3460     \use:e
3461     {
3462         \exp_not:N\AddToHookNext{shipout/foreground}
3463         {
3464             \exp_not:N\int_compare:nNnT
3465             {\exp_not:N\g_shipout_readonly_int}
3466             =
3467             {\property_ref:een{\l__luamplib_BBox_label_tl}{abspage}{0}}
3468             {
3469                 \exp_not:N\put
3470                 (\l__luamplib_BBox_llx_tl bp, \dim_eval:n{\l__luamplib_BBox_lly_tl bp -\paperheight})
3471                 {
3472                     \exp_not:N\opacity_select:n{0.5} \exp_not:N\color_select:n{red}
3473                     \exp_not:N\rule
3474                     {\dim_eval:n {\l__luamplib_BBox_urx_tl bp - \l__luamplib_BBox_llx_tl bp}}
3475                     {\dim_eval:n {\l__luamplib_BBox_ury_tl bp - \l__luamplib_BBox_lly_tl bp}}
3476                 }
3477             }
3478         }
3479     }
3480 }
3481 }
3482 }
3483 \cs_set_nopar:Npn \mplibstarttoPDF #1 #2 #3 #4
3484 {
3485     \prependtomplibbox
3486     \hbox dir TLT\bgroup
3487     \luamplibtaggingbegin % begin tagging
3488     \xdef\MPllx{#1}\xdef\MPlly{#2}%
3489     \xdef\MPurx{#3}\xdef\MPury{#4}%
3490     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
3491     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
3492     \parskip0pt
3493     \leftskip0pt
3494     \parindent0pt
3495     \everypar{}%
3496     \setbox\mplibscratchbox\vbox\bgroup
3497     \SuspendTagging{luamplib.mplibtopdf}% stop tag inside figure
3498     \noindent
3499 }
3500 \cs_set_nopar:Npn \mplibstoptoPDF
3501 {
3502     \par
3503     \egroup
3504     \setbox\mplibscratchbox\hbox
3505     {\hskip-\MPllx bp
3506      \raise-\MPlly bp
3507      \box\mplibscratchbox}%
3508     \setbox\mplibscratchbox\vbox to \MPheight
3509     {\vfill
3510      \hsize\MPwidth

```

```

3511     \wd\mplibscratchbox0pt
3512     \ht\mplibscratchbox0pt
3513     \dp\mplibscratchbox0pt
3514     \box\mplibscratchbox}%
3515 \wd\mplibscratchbox\MPwidth
3516 \ht\mplibscratchbox\MPheight
3517 \luamplibtaggingBBox % BBox
3518 \box\mplibscratchbox
3519 \luamplibtaggingend % end tagging
3520 \egroup
3521 }
3522 \RenewDocumentCommand\mplibcode{0{}}
3523 {
3524   \msg_set:nnn {luamplib}{alt-text-missing}
3525   {
3526     Alternative~text~for~mplibcode~is~missing.\\
3527     Using~the~default~value~'##1'~instead.
3528   }
3529   \tl_gset_eq:NN \currentmpinstancename \c_empty_tl
3530   \keys_set:nn{luamplib/tag}{#1}
3531   \tl_if_empty:NF \currentmpinstancename
3532   { \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure~\currentmpinstancename} }
3533   \mplibtmp toks{}\ltxdomplibcode
3534 }
3535 \RenewDocumentCommand\mpfig{s 0{}}
3536 {
3537   \begingroup
3538   \IfBooleanTF{#1}
3539   {\mplibprempfig *}
3540   {
3541     \msg_set:nnn {luamplib}{alt-text-missing}
3542     {
3543       Alternative~text~for~mpfig~is~missing.\\
3544       Using~the~default~value~'##1'~instead.
3545     }
3546     \keys_set:nn{luamplib/tag}{#2}
3547     \tl_if_empty:NF \mpfiginstancename
3548     { \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure~\mpfiginstancename} }
3549     \mplibmainmpfig
3550   }
3551 }
3552 \RenewDocumentCommand\usemplibgroup{0{ } m}
3553 {
3554   \begingroup
3555   \msg_set:nnn {luamplib}{alt-text-missing}
3556   {
3557     Alternative~text~for~usemplibgroup~is~missing.\\
3558     Using~the~default~value~'##1'~instead.
3559   }
3560   \keys_set:nn{luamplib/tag}{#1}
3561   \tl_set:Nn\l__luamplib_tag_alt_dflt_tl {metapost~figure~#2}
3562   \csname luamplib.group.#2\endcsname
3563   \endgroup
3564 }

```

```
3565 \cs_new_nopar:Npn \mplibalttext #1
3566 {
3567   \tl_set:Nc \l__luamplib_tag_alt_tl {\text_purify:n{#1}}
3568 }
3569 \cs_new_nopar:Npn \mplibactualtext #1
3570 {
3571   \tl_set:Nc \l__luamplib_tag_actual_tl {\text_purify:n{#1}}
3572 }
3573 \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 your rights to do these things. To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

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

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

- This License applies to any program or other work which contains a notice placed by the copyright holder 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.
- You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program. You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
- You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
 - If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

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

on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

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

- You may copy and distribute the Program for a work based on it, under Section 1, 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 binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

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

- You may not copy, modify, sublicense, or distribute the Program except as expressly permitted under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
- You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

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

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

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

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

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

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

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

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

NO WARRANTY

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

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

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

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

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

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

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

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

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

Also add information on how to contact you by electronic and paper mail. If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

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

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

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

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

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

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