

The luamplib package

Hans Hagen, Taco Hoekwater, Elie Roux, Philipp Gesang and Kim Dohyun
Maintainer: LuaLaTeX Maintainers — Support: <lualatex-dev@tug.org>

2023/08/21 v2.25.1

Abstract

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

1 Documentation

This packages 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 (in Lua) for the Lua mplib functions and some TeX functions to have the output of the mplib functions in the pdf.

In the past, the package required PDF mode in order to output something. Starting with version 2.7 it works in DVI mode as well, though DVIPDFMx is the only DVI tool currently supported.

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

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 code is 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, new functionalities have been added by Kim Dohyun. The changes are:

- a \LaTeX environment
- all TeX macros start by `mplib`
- use of `luatexbase` for errors, warnings and declaration
- possibility to use `btex ... etex` to typeset TeX code. `texttext()` is a more versatile macro equivalent to `TEX()` from `TEX.mp`. `TEX()` is also allowed and is a synonym of `texttext()`.

N.B. Since v2.5, `btex ... etex` input from external `mp` files will also be processed by `luamplib`.

N.B. Since v2.20, `verbatimtex ... etex` from external `mp` files will be also processed by `luamplib`. Warning: This is a change from previous version.

Some more changes and cautions are:

\mplibforcehmode When this macro is declared, every mplibcode 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 define this command with anything suitable before a box.)

\mpliblegacybehavior{enable} By default, `\mpliblegacybehavior{enable}` is already declared, in which case a `verbatimtex ... etex` that comes just before `beginfig()` is not ignored, but the \TeX code will be inserted before the following mplib hbox. Using this command, each mplib box can be freely moved horizontally and/or vertically. Also, a box number might be assigned to mplib box, allowing it to be reused later (see test files).

```
\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`.

By contrast, \TeX code in `VerbatimTeX(...)` or `verbatimtex ... etex` between `beginfig()` and `endfig` will be inserted after flushing out the mplib figure.

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

\mpliblegacybehavior{disable} If `\mpliblegacybehavior{disabled}` is declared by user, any `verbatimtex ... etex` will be executed, along with `btex ... etex`, sequentially one by one. So, some \TeX code in `verbatimtex ... etex` will have effects on `btex ... etex` codes that follows.

```
\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}
```

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

\everymplib, \everyendmplib Since v2.3, new macros `\everymplib` and `\everyendmplib` redefine the lua table containing MetaPost code which will be automatically inserted at the beginning and ending of each `mplibcode`.

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

\mpdim Since v2.3, `\mpdim` and other raw \TeX commands are allowed inside `mplib` code. This feature is inspired by `gmp.sty` authored by Enrico Gregorio. Please refer the manual of `gmp` package for details.

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

N.B. Users should not use the protected variant of `btex ... etex` as provided by `gmp` package. As `luamplib` automatically protects \TeX code inbetween, `\btex` is not supported here.

\mpcolor With `\mpcolor` command, color names or expressions of `color/xcolor` packages can be used inside `mplibcode` environment (after `withcolor` operator), though `luamplib` does not automatically load these packages. See the example code above. For spot colors, `(x)spotcolor` (in PDF mode) and `xespotcolor` (in DVI mode) packages are supported as well.

\mplibnumbersystem Users can choose `numbersystem` option since v2.4. The default value `scaled` can be changed to `double` or `decimal` by declaring `\mplibnumbersystem{double}` or `\mplibnumbersystem{decimal}`. For details see <http://github.com/lualatex/luamplib/issues/21>.

Settings regarding cache files To support `btex ... etex` in external `.mp` files, `luamplib` inspects the content of each and every `.mp` input files and makes caches if necessary, before returning their paths to $\text{Lua}\TeX$'s `mplib` library. This would make the compilation time longer wastefully, as most `.mp` files do not contain `btex ... etex` command. So `luamplib` provides macros as follows, so that users can give instruction about files that do not require this functionality.

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

where `<filename>` is a file name 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, in the same directory as where `pdf/dvi` output file is saved. This however can be changed 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.

\mplibtextlabel Starting with v2.6, `\mplibtextlabel{enable}` enables string 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. Every string label therefore will be typeset with current \TeX font. Also take care of `char` operator in the left side argument, as this might bring unpermitted characters into \TeX .

\mplibcodeinherit Starting with v2.9, `\mplibcodeinherit{enable}` enables the inheritance of variables, constants, and macros defined by previous `mplibcode` chunks. On the contrary, the default value `\mplibcodeinherit{disable}` will make each code chunks being treated as an independent instance, and never affected by previous code chunks.

Separate instances for \LaTeX environment v2.22 has added the support for several named MetaPost instances in \LaTeX `mplibcode` environment. Syntax is like so:

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

Behaviour 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` labels still exist separately and require `\mplibglobaltexttext`.
- When an instance names is set, respective `\currentmpinstancename` is set.

In parallel with this functionality, v2.23 and after supports 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`. Syntax is:

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

\mplibglobaltexttext To inherit `btex ... etex` labels as well as metapost variables, it is necessary to declare `\mplibglobaltexttext{enable}` in advance. On this case, be careful that normal \TeX boxes can conflict with `btex ... etex` boxes, though this would occur very rarely. Notwithstanding the danger, it is a ‘must’ option to activate `\mplibglobaltexttext` if you want to use `graph.mp` with `\mplibcodeinherit` functionality.

```
\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 Starting with v2.11, users can issue `\mplibverbatim{enable}`, after which the contents of `mplibcode` environment will be read verbatim. As a result, except for `\mpdim` and `\mpcolor`, all other \TeX commands outside `btex ... etex` or `verbatimex ... etex` are not expanded and will be fed literally into the `mplib` process.

\mplibshowlog When `\mplibshowlog{enable}` is declared, log messages returned by `mplib` instance will be printed into the `.log` file. `\mplibshowlog{disable}` will revert this functionality. This is a \TeX side interface for `luamplib.showlog`. (v2.20.8)

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` or `\mplibforcehmode` are suitable for going into this file.

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>}`.

2 Implementation

2.1 Lua module

```

1
2 luatexbase.provides_module {
3   name      = "luamplib",
4   version   = "2.25.1",
5   date      = "2023/08/21",
6   description = "Lua package to typeset Metapost with LuaTeX's MPLib.",
7 }
8
9 local format, abs = string.format, math.abs
10
11 local err = function(...)
12   return luatexbase.module_error ("luamplib", select("#",...) > 1 and format(...) or ...)
13 end
14 local warn = function(...)
15   return luatexbase.module_warning("luamplib", select("#",...) > 1 and format(...) or ...)
16 end
17 local info = function(...)
18   return luatexbase.module_info ("luamplib", select("#",...) > 1 and format(...) or ...)
19 end
20

```

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

```

21 luamplib      = luamplib or { }
22 local luamplib = luamplib

```

```

23
24 luamplib.showlog = luamplib.showlog or false
25

```

This module is a stripped down version of libraries that are used by ConT_EXt. Provide a few “shortcuts” expected by the imported code.

```

26 local tableconcat = table.concat
27 local texsprint   = tex.sprint
28 local textprint   = tex.tprint
29
30 local texget       = tex.get
31 local texgettoks  = tex.gettoks
32 local texgetbox   = tex.getbox
33 local texruntoks  = tex.runtoks

```

We don’t use tex.scantoks anymore. See below reagrding tex.runtoks.
 local texscantoks = tex.scantoks

```

34
35 if not texruntoks then
36   err("Your LuaTeX version is too old. Please upgrade it to the latest")
37 end
38
39 local mplib = require ('mplib')
40 local kpse  = require ('kpse')
41 local lfs   = require ('lfs')
42
43 local lfsattributes = lfs.attributes
44 local lfsisdir     = lfs.isdir
45 local lfsmkdir     = lfs.mkdir
46 local lfstouch     = lfs.touch
47 local iioopen      = io.open
48

```

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

```

49 local file = file or { }
50 local replacesuffix = file.replacesuffix or function(filename, suffix)
51   return (filename:gsub("%.[%a%d]+$", "")) .. "." .. suffix
52 end
53
54 local is_writable = file.is_writable or function(name)
55   if lfsisdir(name) then
56     name = name .. "/_luam_plib_temp_file_"
57     local fh = iioopen(name,"w")
58     if fh then
59       fh:close(); os.remove(name)
60       return true
61     end
62   end
63 end
64 local mk_full_path = lfs.mkdir or function(path)
65   local full = ""
66   for sub in path:gmatch("(/*[^\w/]+)") do
67     full = full .. sub
68     lfsmkdir(full)

```

```

69 end
70 end
71

```

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.

```

72 local luamplibtime = kpse.find_file("luamplib.lua")
73 luamplibtime = luamplibtime and lfsattributes(luamplibtime,"modification")
74
75 local currenttime = os.time()
76
77 local outputdir
78 if lfstouch then
79   local texmfvar = kpse.expand_var('$TEXMFVAR')
80   if texmfvar and texmfvar ~= "" and texmfvar ~= '$TEXMFVAR' then
81     for _,dir in next, texmfvar:explode(os.type == "windows" and "," or ":") do
82       if not lfsisdir(dir) then
83         mk_full_path(dir)
84       end
85       if is_writable(dir) then
86         local cached = format("%s/luamplib_cache",dir)
87         lfsmkdir(cached)
88         outputdir = cached
89         break
90       end
91     end
92   end
93 end
94 if not outputdir then
95   outputdir = "."
96   for _,v in ipairs(arg) do
97     local t = v:match("%-output%-directory=(.+)")
98     if t then
99       outputdir = t
100       break
101     end
102   end
103 end
104
105 function luamplib.getcachedir(dir)
106   dir = dir:gsub("#", "#")
107   dir = dir:gsub("^~",
108     os.type == "windows" and os.getenv("UserProfile") or os.getenv("HOME"))
109   if lfstouch and dir then
110     if lfsisdir(dir) then
111       if is_writable(dir) then
112         luamplib.cachedir = dir
113       else
114         warn("Directory '%s' is not writable!", dir)
115       end
116     else
117       warn("Directory '%s' does not exist!", dir)
118     end

```

```

119 end
120 end
121

```

Some basic MetaPost files not necessary to make cache files.

```

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

```

format.mp is much complicated, so specially treated.

```

140 local function replaceformatmp(file,newfile,ofmodify)
141   local fh = ioopen(file,"r")
142   if not fh then return file end
143   local data = fh:read("*all"); fh:close()
144   fh = ioopen(newfile,"w")
145   if not fh then return file end
146   fh:write(
147     "let normalinfont = infont;\n",
148     "primarydef str infont name = rawtexttext(str) enddef;\n",
149     data,
150     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
151     "vardef Fexp_(expr x) = rawtexttext(\"$^{\"&decimal x&\"}$\") enddef;\n",
152     "let infont = normalinfont;\n"
153   ); fh:close()
154   lfstouch(newfile,currenttime,ofmodify)
155   return newfile
156 end
157

```

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

```

158 local name_b = "%f[%a_]"
159 local name_e = "%f[^%a_]"
160 local btex_etex = name_b.."btex"..name_e.."s*(.)s*"..name_b.."etex"..name_e
161 local verbatimetex_etex = name_b.."verbatimetex"..name_e.."s*(.)s*"..name_b.."etex"..name_e
162
163 local function replaceinputmpfile (name,file)
164   local ofmodify = lfsattributes(file,"modification")
165   if not ofmodify then return file end
166   local cachedir = luamplib.cachedir or outputdir
167   local newfile = name:gsub("%W","_")

```



```

168 newfile = cachedir ../luamplib_input_..newfile
169 if newfile and luamplibtime then
170   local nf = lfsattributes(newfile)
171   if nf and nf.mode == "file" and
172     ofmodify == nf.modification and luamplibtime < nf.access then
173     return nf.size == 0 and file or newfile
174   end
175 end
176
177 if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
178
179 local fh = ioopen(file,"r")
180 if not fh then return file end
181 local data = fh:read("*all"); fh:close()
182

```

“etex” must be followed by a space or semicolon as specified in LuaTeX manual, which is not the case of standalone MetaPost though.

```

183 local count,cnt = 0,0
184 data, cnt = data:gsub(btex_etex, "btex %1 etex ") -- space
185 count = count + cnt
186 data, cnt = data:gsub(verbatim_etex, "verbatim %1 etex;") -- semicolon
187 count = count + cnt
188
189 if count == 0 then
190   noneedtoreplace[name] = true
191   fh = ioopen(newfile,"w");
192   if fh then
193     fh:close()
194     lfstouch(newfile,currenttime,ofmodify)
195   end
196   return file
197 end
198
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 it 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
215 local special_ftype = {
216   pfb = "type1 fonts",

```

```

217 enc = "enc files",
218 }
219
220 local function finder(name, mode, ftype)
221   if mode == "w" then
222     if name and name ~= "mpout.log" then
223       kpse.record_output_file(name) -- recorder
224     end
225     return name
226   else
227     ftype = special_ftype[ftype] or ftype
228     local file = mpkpse:find_file(name, ftype)
229     if file then
230       if lfstouch and ftype == "mp" and not noneedtoreplace[name] then
231         file = replaceinputmpfile(name, file)
232       end
233     else
234       file = mpkpse:find_file(name, name:match("%a+$"))
235     end
236     if file then
237       kpse.record_input_file(file) -- recorder
238     end
239     return file
240   end
241 end
242 luamplib.finder = finder
243

```

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.)

```

244 if tonumber(mplib.version()) <= 1.50 then
245   err("luamplib no longer supports mplib v1.50 or lower. "..
246     "Please upgrade to the latest version of LuaTeX")
247 end
248
249 local preamble = [[
250   boolean mplib ; mplib := true ;
251   let dump = endinput ;
252   let normalfontsize = fontsize;
253   input %s ;
254 ]]
255
256 local logatload
257 local function reporterror (result, indeed)
258   if not result then
259     err("no result object returned")
260   else
261     local t, e, l = result.term, result.error, result.log
262
263     log has more information than term, so log first (2021/08/02)
264     local log = l or t or "no-term"
265     log = log:gsub("%(Please type a command or say 'end'%)", ""):gsub("\n+", "\n")
266     if result.status > 0 then
267       warn(log)
268     end
269   end
270 end

```

```

266     if result.status > 1 then
267         err(e or "see above messages")
268     end
269     elseif indeed then
270         local log = logatload..log

```

v2.6.1: now `luamplib` does not disregard `show` command, even when `luamplib.showlog` is false. Incidentally, it does not raise error but just prints a warning, even if output has no figure.

```

271     if log:find"\n>>" then
272         warn(log)
273     elseif log:find"%g" then
274         if luamplib.showlog then
275             info(log)
276         elseif not result.fig then
277             info(log)
278         end
279     end
280     logatload = ""
281 else
282     logatload = log
283 end
284 return log
285 end
286 end
287
288 local function luamplibload (name)
289     local mpx = mplib.new {
290         ini_version = true,
291         find_file   = luamplib.finder,

```

Make use of `make_text` and `run_script`, which will co-operate with LuaTeX's `tex.runtoks`. And we provide `numbersystem` option since v2.4. Default value "scaled" can be changed by declaring `\mplibnumbersystem{double}` or `\mplibnumbersystem{decimal}`. See <https://github.com/lualatex/luamplib/issues/21>.

```

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

```

Append our own MetaPost preamble to the preamble above.

```

299 local preamble = preamble .. luamplib.mplibcodepreamble
300 if luamplib.legacy_verbatimtex then
301     preamble = preamble .. luamplib.legacyverbatimmpreamble
302 end
303 if luamplib.texttextlabel then
304     preamble = preamble .. luamplib.texttextlabelpreamble
305 end
306 local result
307 if not mpx then
308     result = { status = 99, error = "out of memory"}
309 else

```

```

310 result = mpx:execute(format(preamble, replacesuffix(name,"mp")))
311 end
312 reporterror(result)
313 return mpx, result
314 end
315

```

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

```

316 local currentformat = "plain"
317
318 local function setformat (name)
319   currentformat = name
320 end
321 luamplib.setformat = setformat
322

```

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

```

323 local function process_indeed (mpx, data)
324   local converted, result = false, {}
325   if mpx and data then
326     result = mpx:execute(data)
327     local log = reporterror(result, true)
328     if log then
329       if result.fig then
330         converted = luamplib.convert(result)
331       else
332         warn("No figure output. Maybe no beginfig/endfig")
333       end
334     end
335   else
336     err("Mem file unloadable. Maybe generated with a different version of mplib?")
337   end
338   return converted, result
339 end
340

```

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

```

341 luamplib.codeinherit = false
342 local mplibinstances = {}
343
344 local function process (data, instancename)

```

The workaround of issue #70 seems to be unnecessary, as we use `make_text` now.

```

if not data:find(name_b.."beginfig%s*%([%+-%s]*d[%.%d%s]*%)" then
  data = data .. "beginfig(-1);endfig;"
end

```

```

345 local defaultinstancename = currentformat .. (luamplib.numbersystem or "scaled")
346   .. tostring(luamplib.texttextlabel) .. tostring(luamplib.legacy_verbatimtex)
347 local currfmt = instancename or defaultinstancename
348 if #currfmt == 0 then
349   currfmt = defaultinstancename
350 end
351 local mpx = mplibinstances[currfmt]
352 local standalone = false

```

```

353 if currfmt == defaultinstancename then
354   standalone = not luamplib.codeinherit
355 end
356 if mpx and standalone then
357   mpx:finish()
358 end
359 if standalone or not mpx then
360   mpx = luamplibload(currentformat)
361   mplibinstances[currfmt] = mpx
362 end
363 return process_indeed(mpx, data)
364 end
365

```

make_text and some run_script uses Lua \TeX 's tex.runtoks, which made possible running \TeX code snippets inside \directlua.

```

366 local catlatex = luatexbase.registernumber("catcodetable@latex")
367 local catat11 = luatexbase.registernumber("catcodetable@atletter")
368

```

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

```

local function run_tex_code_no_use (str, cat)
  cat = cat or catlatex
  texscantoks("mplibtmptoks", cat, str)
  texruntoks("mplibtmptoks")
end

```

```

369 local function run_tex_code (str, cat)
370   cat = cat or catlatex
371   texruntoks(function() texsprint(cat, str) end)
372 end
373

```

Indefinite number of boxes are needed for btex ... etex. So starts at somewhat huge number of box registry. Of course, this may conflict with other packages using many many boxes. (When codeinherit feature is enabled, boxes must be globally defined.) But I don't know any reliable way to escape this danger.

```

374 local tex_box_id = 2047

```

For conversion of sp to bp.

```

375 local factor = 65536*(7227/7200)
376
377 local textext_fmt = [[image(addto currentpicture doublepath unitsquare )].
378 [[xscaled %f yscaled %f shifted (0,-%f) ]].
379 [[withprescript "mplibtexboxid=%i:%f:%f"]]]
380
381 local function process_tex_text (str)
382   if str then
383     tex_box_id = tex_box_id + 1
384     local global = luamplib.globaltextext and "\\global" or ""
385     run_tex_code(format("%s\\setbox%i\\hbox{%s}", global, tex_box_id, str))
386     local box = texgetbox(tex_box_id)

```

```

387 local wd = box.width / factor
388 local ht = box.height / factor
389 local dp = box.depth / factor
390 return texttext_fmt:format(wd, ht+dp, dp, tex_box_id, wd, ht+dp)
391 end
392 return ""
393 end
394

```

Make `color` or `xcolor`'s color expressions usable, with `\mpcolor` or `mplibcolor`. These commands should be used with graphical objects.

```

395 local mpplibcolor_fmt = [[\begingroup\let\XC@color\relax]]..
396 [[\def\set@color{\global\mplibtmp toks\expandafter{\current@color}}]]..
397 [[\color %s \endgroup]]
398
399 local function process_color (str)
400 if str then
401 if not str:find("%.") then
402 str = format("%s",str)
403 end
404 run_tex_code(mplibcolor_fmt:format(str), catat11)
405 return format('1 withprescript "MPLibOverrideColor=%s"', texgettoks"mplibtmp toks")
406 end
407 return ""
408 end
409

```

`\mpdim` is expanded before MPLib process, so code below will not be used for `mplibcode` data. But who knows anyone would want it in `.mp` input file. If then, you can say `mplibdimen(".5\textwidth")` for example.

```

410 local function process_dimen (str)
411 if str then
412 str = str:gsub("{(.+)}", "%1")
413 run_tex_code(format([[mplibtmp toks\expandafter{\the\dimexpr %s\relax}]], str))
414 return format("begingroup %s endgroup", texgettoks"mplibtmp toks")
415 end
416 return ""
417 end
418

```

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

```

419 local function process_verbatimtex_text (str)
420 if str then
421 run_tex_code(str)
422 end
423 return ""
424 end
425

```

For legacy `verbatimtex` process. `verbatimtex ... 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.

```

426 local tex_code_pre_mplib = {}
427 luamplib.figid = 1

```

```

428 luamplib.in_the_fig = false
429
430 local function legacy_mplibcode_reset ()
431   tex_code_pre_mplib = {}
432   luamplib.figid = 1
433 end
434
435 local function process_verbatimtex_prefig (str)
436   if str then
437     tex_code_pre_mplib[luamplib.figid] = str
438   end
439   return ""
440 end
441
442 local function process_verbatimtex_infig (str)
443   if str then
444     return format('special "postmplibverbtex=%s";', str)
445   end
446   return ""
447 end
448
449 local runscript_funcs = {
450   luamplibtext    = process_tex_text,
451   luamplibcolor   = process_color,
452   luamplibdimen   = process_dimen,
453   luamplibprefig  = process_verbatimtex_prefig,
454   luamplibinfig   = process_verbatimtex_infig,
455   luamplibverbtex = process_verbatimtex_text,
456 }
457

```

For metafun format. see issue #79.

```

458 mp = mp or {}
459 local mp = mp
460 mp.mf_path_reset = mp.mf_path_reset or function() end
461 mp.mf_finish_saving_data = mp.mf_finish_saving_data or function() end
462

```

metafun 2021-03-09 changes crashes luamplib.

```

463 catcodes = catcodes or {}
464 local catcodes = catcodes
465 catcodes.numbers = catcodes.numbers or {}
466 catcodes.numbers.ctxcatcodes = catcodes.numbers.ctxcatcodes or catlatex
467 catcodes.numbers.texcatcodes = catcodes.numbers.texcatcodes or catlatex
468 catcodes.numbers.luacatcodes = catcodes.numbers.luacatcodes or catlatex
469 catcodes.numbers.notcatcodes = catcodes.numbers.notcatcodes or catlatex
470 catcodes.numbers.vrbcatcodes = catcodes.numbers.vrbcatcodes or catlatex
471 catcodes.numbers.prtcatcodes = catcodes.numbers.prtcatcodes or catlatex
472 catcodes.numbers.txtcatcodes = catcodes.numbers.txtcatcodes or catlatex
473

```

A function from Con \TeX t general.

```

474 local function mpprint(buffer,...)
475   for i=1,select("#",...) do
476     local value = select(i,...)

```

```

477   if value ~= nil then
478     local t = type(value)
479     if t == "number" then
480       buffer[#buffer+1] = format("%.16f",value)
481     elseif t == "string" then
482       buffer[#buffer+1] = value
483     elseif t == "table" then
484       buffer[#buffer+1] = "(" .. tableconcat(value,",") .. ")"
485     else -- boolean or whatever
486       buffer[#buffer+1] = tostring(value)
487     end
488   end
489 end
490 end
491
492 function luamplib.runscript (code)
493   local id, str = code:match("(.-){(.*)}")
494   if id and str then
495     local f = runscript_funcs[id]
496     if f then
497       local t = f(str)
498       if t then return t end
499     end
500   end
501   local f = loadstring(code)
502   if type(f) == "function" then
503     local buffer = {}
504     function mp.print(...)
505       mpprint(buffer,...)
506     end
507     f()
508     buffer = tableconcat(buffer)
509     if buffer and buffer ~= "" then
510       return buffer
511     end
512     buffer = {}
513     mpprint(buffer, f())
514     return tableconcat(buffer)
515   end
516   return ""
517 end
518

```

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

```

519 local function protecttexcontents (str)
520   return str:gsub("\\\\%", "\\0PerCent\0")
521         :gsub("%%. -\n", "")
522         :gsub("%%. -$", "")
523         :gsub("%zPerCent%", "\\0PerCent")
524         :gsub("%s+", " ")
525 end
526
527 luamplib.legacy_verbatimtex = true
528
529 function luamplib.maketext (str, what)

```



```

530 if str and str ~= "" then
531   str = protecttexcontents(str)
532   if what == 1 then
533     if not str:find("\\documentclass"..name_e) and
534       not str:find("\\begin%s*{document}") and
535       not str:find("\\documentstyle"..name_e) and
536       not str:find("\\usepackage"..name_e) then
537       if luamplib.legacy_verbatimtex then
538         if luamplib.in_the_fig then
539           return process_verbatimtex_infig(str)
540         else
541           return process_verbatimtex_prefig(str)
542         end
543       else
544         return process_verbatimtex_text(str)
545       end
546     end
547   else
548     return process_tex_text(str)
549   end
550 end
551 return ""
552 end
553

```

Our MetaPost preambles

```

554 local mplibcodepreamble = [[
555 texscriptmode := 2;
556 def rawtexttext (expr t) = runscript("luamplibtext{"&t&}") enddef;
557 def mplibcolor (expr t) = runscript("luamplibcolor{"&t&}") enddef;
558 def mplibdimen (expr t) = runscript("luamplibdimen{"&t&}") enddef;
559 def VerbatimTeX (expr t) = runscript("luamplibverbtext{"&t&}") enddef;
560 if known context_mlib:
561   defaultfont := "cmtt10";
562   let infont = normalinfont;
563   let fontsize = normalfontsize;
564   vardef thelabel@#(expr p,z) =
565     if string p :
566       thelabel@#(p infont defaultfont scaled defaultscale,z)
567     else :
568       p shifted (z + labeloffset*mfun_laboff@# -
569         (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +
570         (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
571     fi
572   enddef;
573   def graphicstext primary filename =
574     if (readfrom filename = EOF):
575       errmessage "Please prepare "&filename&" in advance with"&
576         " 'pstoedit -ssp -dt -f mpost yourfile.ps "&filename&"";
577     fi
578     closefrom filename;
579     def data_mpy_file = filename enddef;
580     mfun_do_graphic_text (filename)
581   enddef;
582 else:

```

```

583   vardef texttext@# (text t) = rawtexttext (t) enddef;
584 fi
585 def externalfigure primary filename =
586   draw rawtexttext("\includegraphics{"& filename &}")
587 enddef;
588 def TEX = texttext enddef;
589 ]]
590 luamplib.mplibcodepreamble = mplibcodepreamble
591
592 local legacyverbatimpreamble = [[
593 def specialVerbatimTeX (text t) = runscript("luamplibprefig{"&t&}") enddef;
594 def normalVerbatimTeX (text t) = runscript("luamplibinfig{"&t&}") enddef;
595 let VerbatimTeX = specialVerbatimTeX;
596 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;"&
597   "runscript(" &ditto& "luamplib.in_the_fig=true" &ditto& ");";
598 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;"&
599   "runscript(" &ditto&
600   "if luamplib.in_the_fig then luamplib.figid=luamplib.figid+1 end "&
601   "luamplib.in_the_fig=false" &ditto& ");";
602 ]]
603 luamplib.legacyverbatimpreamble = legacyverbatimpreamble
604
605 local texttextlabelpreamble = [[
606 primarydef s infont f = rawtexttext(s) enddef;
607 def fontsize expr f =
608   begingroup
609     save size; numeric size;
610     size := mplibdimen("1em");
611     if size = 0: 10pt else: size fi
612   endgroup
613 enddef;
614 ]]
615 luamplib.texttextlabelpreamble = texttextlabelpreamble
616

```

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

```

617 luamplib.verbatiminput = false
618

```

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

```

619 local function protect_expansion (str)
620   if str then
621     str = str:gsub("\\", "!!!Control!!!")
622           :gsub("%%", "!!!Comment!!!")
623           :gsub("#", "!!!HashSign!!!")
624           :gsub("{", "!!!LBrace!!!")
625           :gsub("}", "!!!RBrace!!!")
626     return format("\unexpanded{%s}", str)
627   end
628 end
629
630 local function unprotect_expansion (str)
631   if str then
632     return str:gsub("!!!Control!!!", "\\")
633           :gsub("!!!Comment!!!", "%")

```

```

634         :gsub("!!!HashSign!!!", "#")
635         :gsub("!!!LBrace!!!", "{")
636         :gsub("!!!RBrace!!!", "}")
637     end
638 end
639
640 luamplib.everymplib = { ["" ] = "" }
641 luamplib.everyendmplib = { ["" ] = "" }
642
643 local function process_mplibcode (data, instancename)
    This is needed for legacy behavior regarding verbatimex
644     legacy_mplibcode_reset()
645
646     local everymplib = luamplib.everymplib[instancename] or
647                       luamplib.everymplib[""]
648     local everyendmplib = luamplib.everyendmplib[instancename] or
649                           luamplib.everyendmplib[""]
650     data = format("\n%s\n%s\n%s\n", everymplib, data, everyendmplib)
651     data = data:gsub("\r", "\n")
652
653     data = data:gsub("\mpcolor%s+(.-%b{})", "mplibcolor(\%"1\%)")
654     data = data:gsub("\mpdim%s+(%b{})", "mplibdimen(\%"1\%)")
655     data = data:gsub("\mpdim%s+(\%a+)", "mplibdimen(\%"1\%)")
656
657     data = data:gsub(btex_etex, function(str)
658         return format("btex %s etex ", -- space
659             luamplib.verbatiminput and str or protect_expansion(str))
660     end)
661     data = data:gsub(verbatimex_etex, function(str)
662         return format("verbatimex %s etex;", -- semicolon
663             luamplib.verbatiminput and str or protect_expansion(str))
664     end)
665

```

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.

```

666     if not luamplib.verbatiminput then
667         if luamplib.texttextlabel then
668             data = data:gsub("\.-\\"", protect_expansion)
669         end
670
671         data = data:gsub("\\%", "\0PerCent\0")
672         data = data:gsub("%%.-\n", "")
673         data = data:gsub("%zPerCent%z", "\%")
674
675         run_tex_code(format("\mplibmptoks\expanded{{{s}}}", data))
676         data = texgettoks"mplibmptoks"

```

Next line to address issue #55

```

677     data = data:gsub("##", "#")
678     if luamplib.texttextlabel then
679         data = data:gsub("\.-\\"", unprotect_expansion)
680     end
681     data = data:gsub(btex_etex, function(str)

```

```

682     return format("btex %s etex", unprotect_expansion(str))
683   end)
684   data = data:gsub(verbatim_tex_etex, function(str)
685     return format("verbatim_tex %s etex", unprotect_expansion(str))
686   end)
687 end
688
689 process(data, instancename)
690 end
691 luamplib.process_mplibcode = process_mplibcode
692

```

For parsing prescript materials.

```

693 local further_split_keys = {
694   mplibtexboxid = true,
695   sh_color_a    = true,
696   sh_color_b    = true,
697 }
698
699 local function script2table(s)
700   local t = {}
701   for _,i in ipairs(s:explode("\13+")) do
702     local k,v = i:match("(.-)=(.*)") -- v may contain = or empty.
703     if k and v and k ~= "" then
704       if further_split_keys[k] then
705         t[k] = v:explode(":")
706       else
707         t[k] = v
708       end
709     end
710   end
711   return t
712 end
713

```

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

```

714 local function getobjects(result, figure, f)
715   return figure:objects()
716 end
717
718 local function convert(result, flusher)
719   luamplib.flush(result, flusher)
720   return true -- done
721 end
722 luamplib.convert = convert
723
724 local function pdf_startfigure(n,llx,lly,urx,ury)
725   texsprint(format("\mplibstarttoPDF{%f}{%f}{%f}{%f}",llx,lly,urx,ury))
726 end
727
728 local function pdf_stopfigure()
729   texsprint("\mplibstoptoPDF")
730 end
731

```

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

```

732 local function pdf_literalcode(fmt,...) -- table
733   textprint({"\mplibtoPDF"},{-2,format(fmt,...)},{})
734 end
735
736 local function pdf_textfigure(font,size,text,width,height,depth)
737   text = text:gsub(".",function(c)
738     return format("\hbox{\char%i}",string.byte(c)) -- kerning happens in metapost
739   end)
740   texsprint(format("\mplibtexttext{%s}{%f}{%s}{%s}{%f}",font,size,text,0,-( 7200/ 7227)/65536*depth))
741 end
742
743 local bend_tolerance = 131/65536
744
745 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
746
747 local function pen_characteristics(object)
748   local t = mplib.pen_info(object)
749   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
750   divider = sx*sy - rx*ry
751   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
752 end
753
754 local function concat(px, py) -- no tx, ty here
755   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
756 end
757
758 local function curved(ith,pth)
759   local d = pth.left_x - ith.right_x
760   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
761     d = pth.left_y - ith.right_y
762     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
763       return false
764     end
765   end
766   return true
767 end
768
769 local function flushnormalpath(path,open)
770   local pth, ith
771   for i=1,#path do
772     pth = path[i]
773     if not ith then
774       pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
775     elseif curved(ith,pth) then
776       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)
777     else
778       pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
779     end
780     ith = pth
781   end
782   if not open then
783     local one = path[1]

```

```

784   if curved(pth,one) then
785     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 )
786   else
787     pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
788   end
789 elseif #path == 1 then -- special case .. draw point
790   local one = path[1]
791   pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
792 end
793 end
794
795 local function flushconcatpath(path,open)
796 pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
797 local pth, ith
798 for i=1,#path do
799   pth = path[i]
800   if not ith then
801     pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
802   elseif curved(ith,pth) then
803     local a, b = concat(ith.right_x,ith.right_y)
804     local c, d = concat(pth.left_x,pth.left_y)
805     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
806   else
807     pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
808   end
809   ith = pth
810 end
811 if not open then
812   local one = path[1]
813   if curved(pth,one) then
814     local a, b = concat(pth.right_x,pth.right_y)
815     local c, d = concat(one.left_x,one.left_y)
816     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
817   else
818     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
819   end
820 elseif #path == 1 then -- special case .. draw point
821   local one = path[1]
822   pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
823 end
824 end
825
826 local pdfoutput = tonumber(texget("outputmode")) or tonumber(texget("pdfoutput"))
827 local pdfmode = pdfoutput > 0
828
829 local function start_pdf_code()
830   if pdfmode then
831     pdf_literalcode("q")
832   else
833     texsprint("\\special{pdf:bcontent}") -- dvipdfmx
834   end
835 end
836 local function stop_pdf_code()

```

```

837 if pdfmode then
838   pdf_literalcode("Q")
839 else
840   texpstr(" \\special{pdf:econtent}") -- dvipdfmx
841 end
842 end
843

```

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

```

844 local function put_tex_boxes (object,prescript)
845   local box = prescript.mplibtexboxid
846   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
847   if n and tw and th then
848     local op = object.path
849     local first, second, fourth = op[1], op[2], op[4]
850     local tx, ty = first.x_coord, first.y_coord
851     local sx, rx, ry, sy = 1, 0, 0, 1
852     if tw ~= 0 then
853       sx = (second.x_coord - tx)/tw
854       rx = (second.y_coord - ty)/tw
855       if sx == 0 then sx = 0.00001 end
856     end
857     if th ~= 0 then
858       sy = (fourth.y_coord - ty)/th
859       ry = (fourth.x_coord - tx)/th
860       if sy == 0 then sy = 0.00001 end
861     end
862     start_pdf_code()
863     pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
864     texpstr(format("\\mplibputtextbox{i}",n))
865     stop_pdf_code()
866   end
867 end
868

```

Colors and Transparency

```

869 local pdf_objs = {}
870 local token, getpagers, setpagers = newtoken or token
871 local pgf = { bye = "pgfutil@everybye", extgs = "pgf@sys@addpdfresource@extgs@plain" }
872
873 if pdfmode then -- respect luaotfload-colors
874   getpagers = pdf.getpagersources or function() return pdf.pagersources end
875   setpagers = pdf.setpagersources or function(s) pdf.pagersources = s end
876 else
877   texpstr("\\special{pdf:obj @MPLibTr<<>>}",
878         "\\special{pdf:obj @MPLibSh<<>>}")
879 end
880
881 local function update_pdfobjs (os)
882   local on = pdf_objs[os]
883   if on then
884     return on,false
885   end
886 if pdfmode then

```

```

887   on = pdf.immediateobj(os)
888   else
889     on = pdf_objs.cnt or 0
890     pdf_objs.cnt = on + 1
891   end
892   pdf_objs[os] = on
893   return on,true
894 end
895
896 local transparency_modes = { [0] = "Normal",
897   "Normal",      "Multiply",    "Screen",      "Overlay",
898   "SoftLight",   "HardLight",   "ColorDodge", "ColorBurn",
899   "Darken",      "Lighten",     "Difference",  "Exclusion",
900   "Hue",         "Saturation",  "Color",       "Luminosity",
901   "Compatible",
902 }
903
904 local function update_tr_res(res,mode,opaq)
905   local os = format("<</BM /%s/ca %.3f/CA %.3f/AIS false>>",mode,opaq,opaq)
906   local on, new = update_pdfobjs(os)
907   if new then
908     if pdfmode then
909       res = format("%s/MPLibTr%i %i 0 R",res,on,on)
910     else
911       if pgf.loaded then
912         texsprint(format("\csname %s\endcsname{/MPLibTr%i%s}", pgf.extgs, on, os))
913       else
914         texsprint(format("\special{pdf:put @MPLibTr<</MPLibTr%i%s>>}",on,os))
915       end
916     end
917   end
918   return res,on
919 end
920
921 local function tr_pdf_pageresources(mode,opaq)
922   if token and pgf.bye and not pgf.loaded then
923     pgf.loaded = token.create(pgf.bye).cmdname == "assign_toks"
924     pgf.bye    = pgf.loaded and pgf.bye
925   end
926   local res, on_on, off_on = "", nil, nil
927   res, off_on = update_tr_res(res, "Normal", 1)
928   res, on_on  = update_tr_res(res, mode, opaq)
929   if pdfmode then
930     if res ~= "" then
931       if pgf.loaded then
932         texsprint(format("\csname %s\endcsname{%s}", pgf.extgs, res))
933       else
934         local tpr, n = getpageres() or "", 0
935         tpr, n = tpr:gsub("/ExtGState<<", "%1"..res)
936         if n == 0 then
937           tpr = format("%s/ExtGState<<%s>>", tpr, res)
938         end
939         setpageres(tpr)
940       end
941     end
942   end

```



```

941   end
942   else
943     if not pgf.loaded then
944       texsprint(format("\special{pdf:put @resources<</ExtGState @MPLibTr>>}"))
945     end
946   end
947   return on_on, off_on
948 end
949
    Shading with metafun format. (maybe legacy way)
950 local shading_res
951
952 local function shading_initialize ()
953   shading_res = {}
954   if pdfmode and luatexbase.callbacktypes.finish_pdffile then -- ltluatex
955     local shading_obj = pdf.reserveobj()
956     setpagers(format("%s/Shading %i 0 R", getpagers() or "", shading_obj))
957     luatexbase.add_to_callback("finish_pdffile", function()
958       pdf.immediateobj(shading_obj, format("<<s>>", tableconcat(shading_res)))
959       end, "luamplib.finish_pdffile")
960     pdf_objs.finishpdf = true
961   end
962 end
963
964 local function sh_pdfpageresources(shtype, domain, colorspace, colora, colorb, coordinates)
965   if not shading_res then shading_initialize() end
966   local os = format("<</FunctionType 2/Domain [ %s ]/C0 [ %s ]/C1 [ %s ]/N 1>>",
967     domain, colora, colorb)
968   local funcobj = pdfmode and format("%i 0 R", update_pdfobjs(os)) or os
969   os = format("<</ShadingType %i/ColorSpace %s/Function %s/Coords [ %s ]/Extend [ true true ]/AntiAlias true>>",
970     shtype, colorspace, funcobj, coordinates)
971   local on, new = update_pdfobjs(os)
972   if pdfmode then
973     if new then
974       local res = format("/MPLibSh%i %i 0 R", on, on)
975       if pdf_objs.finishpdf then
976         shading_res[#shading_res+1] = res
977       else
978         local pageres = getpagers() or ""
979         if not pageres:find("/Shading<<.*>>") then
980           pageres = pageres.."/Shading<<>>"
981         end
982         pageres = pageres:gsub("/Shading<<","%1"..res)
983         setpagers(pageres)
984       end
985     end
986   else
987     if new then
988       texsprint(format("\special{pdf:put @MPLibSh<</MPLibSh%i%s>>}", on, os))
989     end
990     texsprint(format("\special{pdf:put @resources<</Shading @MPLibSh>>}"))
991   end
992   return on
993 end

```

```

994
995 local function color_normalize(ca,cb)
996   if #cb == 1 then
997     if #ca == 4 then
998       cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
999     else -- #ca = 3
1000       cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
1001     end
1002   elseif #cb == 3 then -- #ca == 4
1003     cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
1004   end
1005 end
1006
1007 local prev_override_color
1008
1009 local function do_preobj_color(object,prescript)
    transparency
1010   local opaq = prescript and prescript.tr_transparency
1011   local tron_no, troff_no
1012   if opaq then
1013     local mode = prescript.tr_alternative or 1
1014     mode = transparency_modes[tonumber(mode)]
1015     tron_no, troff_no = tr_pdf_pageresources(mode,opaq)
1016     pdf_literalcode("/MPlibTr%i gs",tron_no)
1017   end
    color
1018   local override = prescript and prescript.MPlibOverrideColor
1019   if override then
1020     if pdfmode then
1021       pdf_literalcode(override)
1022       override = nil
1023     else
1024       texsprint(format("\special{color push %s}",override))
1025       prev_override_color = override
1026     end
1027   else
1028     local cs = object.color
1029     if cs and #cs > 0 then
1030       pdf_literalcode(luamplib.colorconverter(cs))
1031       prev_override_color = nil
1032     elseif not pdfmode then
1033       override = prev_override_color
1034       if override then
1035         texsprint(format("\special{color push %s}",override))
1036       end
1037     end
1038   end
    shading
1039   local sh_type = prescript and prescript.sh_type
1040   if sh_type then
1041     local domain = prescript.sh_domain
1042     local centera = prescript.sh_center_a:explode()

```

```

1043 local centerb = prescript.sh_center_b:explode()
1044 for _,t in pairs({centera,centerb}) do
1045   for i,v in ipairs(t) do
1046     t[i] = format("%f",v)
1047   end
1048 end
1049 centera = tableconcat(centera," ")
1050 centerb = tableconcat(centerb," ")
1051 local colora = prescript.sh_color_a or {0};
1052 local colorb = prescript.sh_color_b or {1};
1053 for _,t in pairs({colora,colorb}) do
1054   for i,v in ipairs(t) do
1055     t[i] = format("%.3f",v)
1056   end
1057 end
1058 if #colora > #colorb then
1059   color_normalize(colora,colorb)
1060 elseif #colorb > #colora then
1061   color_normalize(colorb,colora)
1062 end
1063 local colorspace
1064 if #colorb == 1 then colorspace = "DeviceGray"
1065 elseif #colorb == 3 then colorspace = "DeviceRGB"
1066 elseif #colorb == 4 then colorspace = "DeviceCMYK"
1067 else return troff_no,override
1068 end
1069 colora = tableconcat(colora, " ")
1070 colorb = tableconcat(colorb, " ")
1071 local shade_no
1072 if sh_type == "linear" then
1073   local coordinates = tableconcat({centera,centerb}," ")
1074   shade_no = sh_pdfpageresources(2,domain,colorspace,colora,colorb,coordinates)
1075 elseif sh_type == "circular" then
1076   local radiusa = format("%f",prescript.sh_radius_a)
1077   local radiusb = format("%f",prescript.sh_radius_b)
1078   local coordinates = tableconcat({centera,radiusa,centerb,radiusb}," ")
1079   shade_no = sh_pdfpageresources(3,domain,colorspace,colora,colorb,coordinates)
1080 end
1081 pdf_literalcode("q /Pattern cs")
1082 return troff_no,override,shade_no
1083 end
1084 return troff_no,override
1085 end
1086
1087 local function do_postobj_color(tr,over,sh)
1088   if sh then
1089     pdf_literalcode("W n /MPLibSh%s sh Q",sh)
1090   end
1091   if over then
1092     texsprintf("\\special{color pop}")
1093   end
1094   if tr then
1095     pdf_literalcode("/MPLibTr%i gs",tr)
1096   end

```

```
1097 end
1098
```

Finally, flush figures by inserting PDF literals.

```
1099 local function flush(result,flusher)
1100   if result then
1101     local figures = result.fig
1102     if figures then
1103       for f=1, #figures do
1104         info("flushing figure %s",f)
1105         local figure = figures[f]
1106         local objects = getobjects(result,figure,f)
1107         local fignum = tonumber(figure:filename():match("[%d]+$") or figure:charcode() or 0)
1108         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1109         local bbox = figure:boundingbox()
1110         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
1111         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()
```

```
1112     else
```

For legacy behavior. Insert ‘pre-fig’ \TeX code here, and prepare a table for ‘in-fig’ codes.

```
1113     if tex_code_pre_mplib[f] then
1114       texsprint(tex_code_pre_mplib[f])
1115     end
1116     local TeX_code_bot = {}
1117     pdf_startfigure(fignum,llx,lly,urx,ury)
1118     start_pdf_code()
1119     if objects then
1120       local savedpath = nil
1121       local savedhtap = nil
1122       for o=1,#objects do
1123         local object      = objects[o]
1124         local objecttype  = object.type
```

The following 5 lines are part of `btex...etex` patch. Again, colors are processed at this stage.

```
1125         local prescript      = object.prescript
1126         prescript = prescript and script2table(prescript) -- prescript is now a table
1127         local tr_opaq,cr_over,shade_no = do_preobj_color(object,prescript)
1128         if prescript and prescript.mplibtexboxid then
1129           put_tex_boxes(object,prescript)
1130         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then --skip
1131         elseif objecttype == "start_clip" then
1132           local evenodd = not object.istext and object.postscript == "evenodd"
1133           start_pdf_code()
1134           flushnormalpath(object.path,false)
1135           pdf_literalcode(evenodd and "W* n" or "W n")
```

```

1136         elseif objecttype == "stop_clip" then
1137             stop_pdf_code()
1138             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1139         elseif objecttype == "special" then
Collect TeX codes that will be executed after flushing. Legacy behavior.
1140             if prescript and prescript.postmplibverbtx then
1141                 TeX_code_bot[#TeX_code_bot+1] = prescript.postmplibverbtx
1142             end
1143         elseif objecttype == "text" then
1144             local ot = object.transform -- 3,4,5,6,1,2
1145             start_pdf_code()
1146             pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
1147             pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.depth)
1148             stop_pdf_code()
1149         else
1150             local evenodd, collect, both = false, false, false
1151             local postscript = object.postscript
1152             if not object.istext then
1153                 if postscript == "evenodd" then
1154                     evenodd = true
1155                 elseif postscript == "collect" then
1156                     collect = true
1157                 elseif postscript == "both" then
1158                     both = true
1159                 elseif postscript == "eoboth" then
1160                     evenodd = true
1161                     both = true
1162                 end
1163             end
1164             if collect then
1165                 if not savedpath then
1166                     savedpath = { object.path or false }
1167                     savedhtap = { object.htap or false }
1168                 else
1169                     savedpath[#savedpath+1] = object.path or false
1170                     savedhtap[#savedhtap+1] = object.htap or false
1171                 end
1172             else
1173                 local ml = object.miterlimit
1174                 if ml and ml ~= miterlimit then
1175                     miterlimit = ml
1176                     pdf_literalcode("%f M",ml)
1177                 end
1178                 local lj = object.linejoin
1179                 if lj and lj ~= linejoin then
1180                     linejoin = lj
1181                     pdf_literalcode("%i j",lj)
1182                 end
1183                 local lc = object.linecap
1184                 if lc and lc ~= linecap then
1185                     linecap = lc
1186                     pdf_literalcode("%i J",lc)
1187                 end
1188                 local dl = object.dash

```

```

1189         if dl then
1190             local d = format("[%s] %f d",tableconcat(dl.dashes or {}, " "),dl.offset)
1191             if d ~= dashed then
1192                 dashed = d
1193                 pdf_literalcode(dashed)
1194             end
1195         elseif dashed then
1196             pdf_literalcode("[ ] 0 d")
1197             dashed = false
1198         end
1199         local path = object.path
1200         local transformed, penwidth = false, 1
1201         local open = path and path[1].left_type and path[#path].right_type
1202         local pen = object.pen
1203         if pen then
1204             if pen.type == 'elliptical' then
1205                 transformed, penwidth = pen_characteristics(object) -- boolean, value
1206                 pdf_literalcode("%f w",penwidth)
1207                 if objecttype == 'fill' then
1208                     objecttype = 'both'
1209                 end
1210             else -- calculated by mplib itself
1211                 objecttype = 'fill'
1212             end
1213         end
1214         if transformed then
1215             start_pdf_code()
1216         end
1217         if path then
1218             if savedpath then
1219                 for i=1,#savedpath do
1220                     local path = savedpath[i]
1221                     if transformed then
1222                         flushconcatpath(path,open)
1223                     else
1224                         flushnormalpath(path,open)
1225                     end
1226                 end
1227                 savedpath = nil
1228             end
1229             if transformed then
1230                 flushconcatpath(path,open)
1231             else
1232                 flushnormalpath(path,open)
1233             end

```

Change from ConTeXt general: there was color stuffs.

```

1234         if not shade_no then -- conflict with shading
1235             if objecttype == "fill" then
1236                 pdf_literalcode(evenodd and "h f*" or "h f")
1237             elseif objecttype == "outline" then
1238                 if both then
1239                     pdf_literalcode(evenodd and "h B*" or "h B")
1240                 else
1241                     pdf_literalcode(open and "S" or "h S")

```

```

1242         end
1243     elseif objecttype == "both" then
1244         pdf_literalcode(evenodd and "h B*" or "h B")
1245     end
1246 end
1247 end
1248 if transformed then
1249     stop_pdf_code()
1250 end
1251 local path = object.htap
1252 if path then
1253     if transformed then
1254         start_pdf_code()
1255     end
1256     if savedhtap then
1257         for i=1,#savedhtap do
1258             local path = savedhtap[i]
1259             if transformed then
1260                 flushconcatpath(path,open)
1261             else
1262                 flushnormalpath(path,open)
1263             end
1264         end
1265         savedhtap = nil
1266         evenodd = true
1267     end
1268     if transformed then
1269         flushconcatpath(path,open)
1270     else
1271         flushnormalpath(path,open)
1272     end
1273     if objecttype == "fill" then
1274         pdf_literalcode(evenodd and "h f*" or "h f")
1275     elseif objecttype == "outline" then
1276         pdf_literalcode(open and "S" or "h S")
1277     elseif objecttype == "both" then
1278         pdf_literalcode(evenodd and "h B*" or "h B")
1279     end
1280     if transformed then
1281         stop_pdf_code()
1282     end
1283 end
1284 end
1285 end

```

Added to ConTeXt general: color stuff. And execute legacy verbatimex code.

```

1286     do_postobj_color(tr_opaq,cr_over,shade_no)
1287 end
1288 end
1289 stop_pdf_code()
1290 pdf_stopfigure()
1291 if #TeX_code_bot > 0 then texsprint(TeX_code_bot) end
1292 end
1293 end
1294 end

```

```

1295 end
1296 end
1297 luamplib.flush = flush
1298
1299 local function colorconverter(cr)
1300   local n = #cr
1301   if n == 4 then
1302     local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
1303     return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
1304   elseif n == 3 then
1305     local r, g, b = cr[1], cr[2], cr[3]
1306     return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
1307   else
1308     local s = cr[1]
1309     return format("%.3f g %.3f G",s,s), "0 g 0 G"
1310   end
1311 end
1312 luamplib.colorconverter = colorconverter

```

2.2 T_EX package

First we need to load some packages.

```

1313 \bgroup\expandafter\expandafter\expandafter\egroup
1314 \expandafter\ifx\csname selectfont\endcsname\relax
1315   \input ltluatex
1316 \else
1317   \NeedsTeXFormat{LaTeX2e}
1318   \ProvidesPackage{luamplib}
1319     [2023/08/21 v2.25.1 mplib package for LuaTeX]
1320   \ifx\newluafunction\undefined
1321     \input ltluatex
1322   \fi
1323 \fi

```

Loading of lua code.

```
1324 \directlua{require("luamplib")}
```

Support older engine. Seems we don't need it, but no harm.

```

1325 \ifx\pdfoutput\undefined
1326   \let\pdfoutput\outputmode
1327   \protected\def\pdfliteral{\pdfextension literal}
1328 \fi

```

Unfortunately there are still packages out there that think it is a good idea to manually set `\pdfoutput` which defeats the above branch that defines `\pdfliteral`. To cover that case we need an extra check.

```

1329 \ifx\pdfliteral\undefined
1330   \protected\def\pdfliteral{\pdfextension literal}
1331 \fi

```

Set the format for metapost.

```
1332 \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 warning.


```

1333 \ifnum\pdfoutput>0
1334 \let\mplibtoPDF\pdfliteral
1335 \else
1336 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
1337 \ifcsname PackageWarning\endcsname
1338 \PackageWarning{luamplib}{take dvipdfmx path, no support for other dvi tools currently.}
1339 \else
1340 \write128{}
1341 \write128{luamplib Warning: take dvipdfmx path, no support for other dvi tools currently.}
1342 \write128{}
1343 \fi
1344 \fi

```

Make `mplibcode` typesetted always in horizontal mode.

```

1345 \def\mplibforcehmode{\let\prependtomplibbox\leavevmode}
1346 \def\mplibnoforcehmode{\let\prependtomplibbox\relax}
1347 \mplibnoforcehmode

```

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

```

1348 \def\mplibsetupcatcodes{%
1349 %catcode'\{=12 %catcode'\}=12
1350 \catcode'\#=12 \catcode'\^=12 \catcode'\~=12 \catcode'\_ =12
1351 \catcode'\&=12 \catcode'\$=12 \catcode'\%=12 \catcode'\^M=12
1352 }

```

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

```

1353 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}

```

The Plain-specific stuff.

```

1354 \unless\ifcsname ver@luamplib.sty\endcsname
1355 \def\mplibcode{%
1356 \begingroup
1357 \begingroup
1358 \mplibsetupcatcodes
1359 \mplibdocode
1360 }
1361 \long\def\mplibdocode#1\endmplibcode{%
1362 \endgroup
1363 \directlua{luamplib.process_mplibcode(===[\unexpanded{#1}]===, ""')}%
1364 \endgroup
1365 }
1366 \else

```

The \LaTeX -specific part: a new environment.

```

1367 \newenvironment{mplibcode}[1][{}]{%
1368 \global\def\currentmpinstancename{#1}%
1369 \mplibtmptoks}\ltxdomplibcode
1370 }{}
1371 \def\ltxdomplibcode{%
1372 \begingroup
1373 \mplibsetupcatcodes
1374 \ltxdomplibcodeindeed
1375 }
1376 \def\mplib@mplibcode{mplibcode}
1377 \long\def\ltxdomplibcodeindeed#1\end#2{%
1378 \endgroup

```

```

1379 \mplibmptoks\expandafter{\the\mplibmptoks#1}%
1380 \def\mplibtemp@a{#2}%
1381 \ifx\mplib@mplibcode\mplibtemp@a
1382 \directlua{luamplib.process_mplibcode(====[\the\mplibmptoks]====,"currentmpinstancename")}%
1383 \end{mplibcode}%
1384 \else
1385 \mplibmptoks\expandafter{\the\mplibmptoks\end{#2}}%
1386 \expandafter\ltxdomplibcode
1387 \fi
1388 }
1389 \fi

```

User settings.

```

1390 \def\mplibshowlog#1{\directlua{
1391   local s = string.lower("#1")
1392   if s == "enable" or s == "true" or s == "yes" then
1393     luamplib.showlog = true
1394   else
1395     luamplib.showlog = false
1396   end
1397 }}
1398 \def\mpliblegacybehavior#1{\directlua{
1399   local s = string.lower("#1")
1400   if s == "enable" or s == "true" or s == "yes" then
1401     luamplib.legacy_verbatimex = true
1402   else
1403     luamplib.legacy_verbatimex = false
1404   end
1405 }}
1406 \def\mplibverbatim#1{\directlua{
1407   local s = string.lower("#1")
1408   if s == "enable" or s == "true" or s == "yes" then
1409     luamplib.verbatiminput = true
1410   else
1411     luamplib.verbatiminput = false
1412   end
1413 }}
1414 \newtoks\mplibmptoks

```

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

```

1415 \protected\def\everymplib{%
1416 \begingroup
1417 \mplibsetupcatcodes
1418 \mplibdoeverymplib
1419 }
1420 \protected\def\everyendmplib{%
1421 \begingroup
1422 \mplibsetupcatcodes
1423 \mplibdoeveryendmplib
1424 }
1425 \ifcsname ver@luamplib.sty\endcsname
1426 \newcommand\mplibdoeverymplib[2][{}]{%
1427 \endgroup
1428 \directlua{
1429   luamplib.everymplib["#1"] = [====[\unexpanded{#2}]====]

```

```

1430 }%
1431 }
1432 \newcommand\mplibdoeveryendmplib[2][]{%
1433 \endgroup
1434 \directlua{
1435   luamplib.everyendmplib["#1"] = [==[\unexpanded{#2}]===]
1436 }%
1437 }
1438 \else
1439 \long\def\mplibdoeverymplib#1{%
1440 \endgroup
1441 \directlua{
1442   luamplib.everymplib[""] = [==[\unexpanded{#1}]===]
1443 }%
1444 }
1445 \long\def\mplibdoeveryendmplib#1{%
1446 \endgroup
1447 \directlua{
1448   luamplib.everyendmplib[""] = [==[\unexpanded{#1}]===]
1449 }%
1450 }
1451 \fi

```

Allow T_EX dimen/color macros. Now runscript does the job, so the following lines are not needed for most cases. But the macros will be expanded when they are used in another macro.

```

1452 \def\mpdim#1{ mplibdimen("#1") }
1453 \def\mpcolor#1#\domplibcolor{#1}}
1454 \def\domplibcolor#1#2{ mplibcolor("#1{#2}") }

```

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

```

1455 \def\mplibnumbersystem#1{\directlua{
1456   local t = "#1"
1457   if t == "binary" then t = "decimal" end
1458   luamplib.numbersystem = t
1459 }}

```

Settings for .mp cache files.

```

1460 \def\mplibmakenocache#1{\mplibdomakenocache #1,*}
1461 \def\mplibdomakenocache#1,{%
1462   \ifx\empty#1\empty
1463     \expandafter\mplibdomakenocache
1464   \else
1465     \ifx*#1\else
1466       \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
1467       \expandafter\expandafter\expandafter\mplibdomakenocache
1468     \fi
1469   \fi
1470 }
1471 \def\mplibcancelnocache#1{\mplibdocancelnocache #1,*}
1472 \def\mplibdocancelnocache#1,{%
1473   \ifx\empty#1\empty
1474     \expandafter\mplibdocancelnocache
1475   \else
1476     \ifx*#1\else

```

```

1477     \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
1478     \expandafter\expandafter\expandafter\mplibdocancelnocache
1479     \fi
1480 \fi
1481 }
1482 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}

```

More user settings.

```

1483 \def\mplibtexttextlabel#1{\directlua{
1484     local s = string.lower("#1")
1485     if s == "enable" or s == "true" or s == "yes" then
1486         luamplib.texttextlabel = true
1487     else
1488         luamplib.texttextlabel = false
1489     end
1490 }}
1491 \def\mplibcodeinherit#1{\directlua{
1492     local s = string.lower("#1")
1493     if s == "enable" or s == "true" or s == "yes" then
1494         luamplib.codeinherit = true
1495     else
1496         luamplib.codeinherit = false
1497     end
1498 }}
1499 \def\mplibglobaltexttext#1{\directlua{
1500     local s = string.lower("#1")
1501     if s == "enable" or s == "true" or s == "yes" then
1502         luamplib.globaltexttext = true
1503     else
1504         luamplib.globaltexttext = false
1505     end
1506 }}

```

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

```

1507 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

```

We encapsulate the literals.

```

1508 \def\mplibstarttoPDF#1#2#3#4{%
1509     \prependtomplibbox
1510     \hbox\bgroup
1511     \xdef\MPllx{#1}\xdef\MPlly{#2}%
1512     \xdef\MPurx{#3}\xdef\MPury{#4}%
1513     \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
1514     \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
1515     \parskip0pt%
1516     \leftskip0pt%
1517     \parindent0pt%
1518     \everypar{}%
1519     \setbox\mplibscratchbox\vbox\bgroup
1520     \noindent
1521 }
1522 \def\mplibstoptoPDF{%
1523     \par
1524     \egroup %
1525     \setbox\mplibscratchbox\hbox %

```

```

1526 {\hskip-\MPllx bp%
1527 \raise-\MPlly bp%
1528 \box\mplibscratchbox}%
1529 \setbox\mplibscratchbox\hbox to \MPheight
1530 {\vfill
1531 \hsize\MPwidth
1532 \wd\mplibscratchbox0pt%
1533 \ht\mplibscratchbox0pt%
1534 \dp\mplibscratchbox0pt%
1535 \box\mplibscratchbox}%
1536 \wd\mplibscratchbox\MPwidth
1537 \ht\mplibscratchbox\MPheight
1538 \box\mplibscratchbox
1539 \egroup
1540 }

```

Text items have a special handler.

```

1541 \def\mplibtexttext#1#2#3#4#5{%
1542 \begingroup
1543 \setbox\mplibscratchbox\hbox
1544 {\font\temp=#1 at #2bp%
1545 \temp
1546 #3}%
1547 \setbox\mplibscratchbox\hbox
1548 {\hskip#4 bp%
1549 \raise#5 bp%
1550 \box\mplibscratchbox}%
1551 \wd\mplibscratchbox0pt%
1552 \ht\mplibscratchbox0pt%
1553 \dp\mplibscratchbox0pt%
1554 \box\mplibscratchbox
1555 \endgroup
1556 }

```

Input luamplib.cfg when it exists.

```

1557 \openin0=luamplib.cfg
1558 \ifeof0 \else
1559 \closein0
1560 \input luamplib.cfg
1561 \fi

```

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 freely redistribute 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 this 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.

- 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.