

# The luamplib package

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

2018/01/04 v2.12.2

## 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 mp`lib` library, that runs metapost code. This package is basically a wrapper (in Lua) for the Lua mp`lib` functions and some TeX functions to have the output of the mp`lib` 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 h`box` 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 mp`lib`code 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 mp`lib`
- 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`. However, `verbatimtex ... etex` will be entirely ignored in this case.

- `verbatimtex ... etex` (in  $\TeX$  file) that comes just before `beginfig()` is not ignored, but the  $\TeX$  code inbetween 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). E.G.

```

\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`.

- $\TeX$  code in `VerbatimTeX(...)` or `verbatimtex ... etex` (in  $\TeX$  file) between `beginfig()` and `endfig` will be inserted after flushing out the `mplib` figure. E.G.

```

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

```

- 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`.
- Since v2.3, new macros `\everymplib` and `\everyendmplib` redefine token lists `\everymplibtoks` and `\everyendmplibtoks` respectively, which will be automatically inserted at the beginning and ending of each `mplib` code. E.G.

```

\everymplib{ verbatimtex \leavevmode etex; beginfig(0); }
\everyendmplib{ endfig; }
\mplibcode % beginfig/endfig not needed; always in horizontal mode
draw fullcircle scaled 1cm;
\endmplibcode

```

N.B. Many users have complained that `mplib` figures do not respect alignment commands such as `\centering` or `\raggedleft`. That's because `luamplib` does not force horizontal or vertical mode. If you want all `mplib` figures center- (or right-) aligned, please use `\everymplib` command with `\leavevmode` as shown above.

- 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. E.G.

```
\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.

- With `\mpcolor` command, color names or expressions of `color/xcolor` packages can be used inside `mplibcode` environment, 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.
- Users can choose `numbersystem` option since v2.4. The default value `scaled` can be changed to `double` by declaring `\mplibnumbersystem{double}`. For details see <http://github.com/lualatex/luamplib/issues/21>.
- 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 LuaTeX'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.
- Starting with v2.6, `\mplibtexttextlabel{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  $\TeX$ .

- 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.

N.B. 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

```

- Starting with v2.11, users can issue `\mplibverbatim{enable}`, after which the contents of `mplibcode` environment will be read verbatim. As a result, users cannot use `\mpdim`, `\mpcolor` etc. All  $\TeX$  commands outside of `btex ... etex` or `verbatimtex ... etex` are not expanded and will be fed literally into the `mplib` process.
- 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 `\mplibcachedir` 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

Use the `luamplib` namespace, since `mplib` is for the metapost library itself. Con<sub>T</sub>E<sub>X</sub>t uses `metapost`.

```
1
2 luamplib          = luamplib or { }
3
```

Identification.

```
4
5 local luamplib    = luamplib
6 luamplib.showlog = luamplib.showlog or false
7 luamplib.lastlog = ""
8
9 luatexbase.provides_module {
10  name          = "luamplib",
11  version       = "2.12.2",
12  date          = "2018/01/04",
13  description   = "Lua package to typeset Metapost with LuaTeX's MPLib.",
14 }
15
```

This module is a stripped down version of libraries that are used by Con<sub>T</sub>E<sub>X</sub>t. Provide a few “shortcuts” expected by the imported code.

```
16
17 local format, abs = string.format, math.abs
18
19 local err = function(...) return luatexbase.module_error ("luamplib", format(...)) end
20 local warn = function(...) return luatexbase.module_warning("luamplib", format(...)) end
21 local info = function(...) return luatexbase.module_info ("luamplib", format(...)) end
22
23 local stringgsub = string.gsub
24 local stringfind = string.find
25 local stringmatch = string.match
26 local stringgmatch = string.gmatch
27 local stringexplode = string.explode
28 local tableconcat = table.concat
29 local textsprint = tex.sprint
30 local textprint = tex.tprint
31
32 local texget = tex.get
33 local texgettoks = tex.gettoks
34 local texgetbox = tex.getbox
35
36 local mplib = require ('mplib')
37 local kpse = require ('kpse')
38 local lfs = require ('lfs')
```

```

39
40 local lfsattributes = lfs.attributes
41 local lfsisdir      = lfs.isdir
42 local lfsmkdir      = lfs.mkdir
43 local lfstouch      = lfs.touch
44 local iioopen       = io.open
45
46 local file = file or { }

```

This is a small trick for  $\LaTeX$ . In  $\LaTeX$  we read the metapost code line by line, but it needs to be passed entirely to `process()`, so we simply add the lines in `data` and at the end we call `process(data)`.

A few helpers, taken from `l-file.lua`.

```

47 local replacesuffix = file.replacesuffix or function(filename, suffix)
48   return (stringgsub(filename,"%.[%a%d]+$", "")) .. "." .. suffix
49 end
50 local stripsuffix = file.stripsuffix or function(filename)
51   return (stringgsub(filename,"%.[%a%d]+$", ""))
52 end
53

```

`btex ... etex` in input `.mp` files will be replaced in `finder`.

```

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.mkdirs or function(path)
65   local full = ""
66   for sub in stringgmach(path,"(/^[^\\/] +)") do
67     full = full .. sub
68     lfsmkdir(full)
69   end
70 end
71
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,stringexplode(texmfvar,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 = stringmatch(v,"%-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 '"..dir.."' is not writable!")
115             end
116         else
117             warn("Directory '"..dir.."' does not exist!")
118         end
119     end
120 end
121
122 local noneedtoreplace = {
123     ["boxes.mp"] = true,
124     -- ["format.mp"] = true,
125     ["graph.mp"] = true,
126     ["marith.mp"] = true,
127     ["mfplain.mp"] = true,
128     ["mpost.mp"] = true,
129     ["plain.mp"] = true,
130     ["rboxes.mp"] = true,
131     ["sarith.mp"] = true,

```

```

132 ["string.mp"] = true,
133 ["TEX.mp"] = true,
134 ["metafun.mp"] = true,
135 ["metafun.mpiv"] = true,
136 ["mp-abck.mpiv"] = true,
137 ["mp-apos.mpiv"] = true,
138 ["mp-asnc.mpiv"] = true,
139 ["mp-bare.mpiv"] = true,
140 ["mp-base.mpiv"] = true,
141 ["mp-butt.mpiv"] = true,
142 ["mp-char.mpiv"] = true,
143 ["mp-chem.mpiv"] = true,
144 ["mp-core.mpiv"] = true,
145 ["mp-crop.mpiv"] = true,
146 ["mp-figs.mpiv"] = true,
147 ["mp-form.mpiv"] = true,
148 ["mp-func.mpiv"] = true,
149 ["mp-grap.mpiv"] = true,
150 ["mp-grid.mpiv"] = true,
151 ["mp-grph.mpiv"] = true,
152 ["mp-idea.mpiv"] = true,
153 ["mp-luas.mpiv"] = true,
154 ["mp-mlib.mpiv"] = true,
155 ["mp-page.mpiv"] = true,
156 ["mp-shap.mpiv"] = true,
157 ["mp-step.mpiv"] = true,
158 ["mp-text.mpiv"] = true,
159 ["mp-tool.mpiv"] = true,
160 }
161 luamplib.noneedtoreplace = noneedtoreplace
162
163 local function replaceformatmp(file,newfile,ofmodify)
164   local fh = ioopen(file,"r")
165   if not fh then return file end
166   local data = fh:read("*all"); fh:close()
167   fh = ioopen(newfile,"w")
168   if not fh then return file end
169   fh:write(
170     "let normalinfont = infont;\n",
171     "primarydef str infont name = rawtexttext(str) enddef;\n",
172     data,
173     "vardef Fmant_(expr x) = rawtexttext(decimal abs x) enddef;\n",
174     "vardef Fexp_(expr x) = rawtexttext("\${\&decimal x\}$\") enddef;\n",
175     "let infont = normalinfont;\n"
176   ); fh:close()
177   lfstouch(newfile,currenttime,ofmodify)
178   return newfile
179 end
180
181 local esctex = "!!!T!!!E!!!X!!!"

```



```

182 local esclbr = "!!!!LEFTBRCE!!!!"
183 local esrbr = "!!!!RGHTBRCE!!!!"
184 local escpcnt = "!!!!PERCENT!!!!"
185 local eschash = "!!!!HASH!!!!"
186 local begname = "%f[A-Z_a-z]"
187 local endname = "%f[^A-Z_a-z]"
188
189 local btex_etex = begname.."btex"..endname.."s*(.)s*"..begname.."etex"..endname
190 local verbatimetex_etex = begname.."verbatimetex"..endname.."s*(.)s*"..begname.."etex"..endname
191
192 local function protecttexcontents(str)
193   return str:gsub("\\\\%", "\\\\"..escpcnt)
194         :gsub("%%.-\n", "")
195         :gsub("%%.-$", "")
196         :gsub("'", "'&ditto'")
197         :gsub("\n%s*", " ")
198         :gsub(escpcnt, "%%")
199 end
200
201 local function replaceinputmpfile (name,file)
202   local ofmodify = lfsattributes(file,"modification")
203   if not ofmodify then return file end
204   local cachedir = luamplib.cachedir or outputdir
205   local newfile = name:gsub("%w", "_")
206   newfile = cachedir .."/luamplib_input_"..newfile
207   if newfile and luamplibtime then
208     local nf = lfsattributes(newfile)
209     if nf and nf.mode == "file" and ofmodify == nf.modification and luamplibtime < nf.access then
210       return nf.size == 0 and file or newfile
211     end
212   end
213   if name == "format.mp" then return replaceformatmp(file,newfile,ofmodify) end
214
215   local fh = ioopen(file,"r")
216   if not fh then return file end
217   local data = fh:read("*all"); fh:close()
218
219   local count,cnt = 0,0
220
221   data = data:gsub("\n[^\n]-\n", function(str)
222     return str:gsub("([bem])tex"..endname,"%1"..escctex)
223   end)
224
225   data, cnt = data:gsub(btex_etex, function(str)
226     return format("rawtexttext(\\"%s\\")",protecttexcontents(str))
227   end)
228   count = count + cnt
229   data, cnt = data:gsub(verbatimetex_etex, "")
230   count = count + cnt
231

```

```

232 data = data:gsub("\^[^\n]-\\"", function(str) -- restore string btex .. etex
233   return str:gsub("([bem])"..esctex, "%1tex")
234 end)
235
236 if count == 0 then
237   needtoreplace[name] = true
238   fh = ioopen(newfile,"w");
239   if fh then
240     fh:close()
241     lfstouch(newfile,currenttime,ofmodify)
242   end
243   return file
244 end
245 fh = ioopen(newfile,"w")
246 if not fh then return file end
247 fh:write(data); fh:close()
248 lfstouch(newfile,currenttime,ofmodify)
249 return newfile
250 end
251
252 local randomseed = nil

```

As the finder function for mplib, use the kpse library and make it behave like as if MetaPost was used (or almost, since the engine name is not set this way—not sure if this is a problem).

```

253
254 local mpkpse = kpse.new("luatex", "mpost")
255
256 local special_ftype = {
257   pfb = "type1 fonts",
258   enc = "enc files",
259 }
260
261 local function finder(name, mode, ftype)
262   if mode == "w" then
263     return name
264   else
265     ftype = special_ftype[ftype] or ftype
266     local file = mpkpse:find_file(name,ftype)
267     if file then
268       if not lfstouch or ftype ~= "mp" or needtoreplace[name] then
269         return file
270       end
271       return replaceinputmpfile(name,file)
272     end
273     return mpkpse:find_file(name,stringmatch(name,"[a-zA-Z]+$"))
274   end
275 end
276 luamplib.finder = finder
277

```

The rest of this module is not documented. More info can be found in the Lua $\TeX$  manual, articles in user group journals and the files that ship with Con $\TeX$ t.

```
278
279 function luamplib.resetlastlog()
280   luamplib.lastlog = ""
281 end
282
```

Below included is section that defines fallbacks for older versions of mplib.

```
283 local mplibone = tonumber(mplib.version()) <= 1.50
284
285 if mplibone then
286
287   luamplib.make = luamplib.make or function(name, mem_name, dump)
288     local t = os.clock()
289     local mpx = mplib.new {
290       ini_version = true,
291       find_file = luamplib.finder,
292       job_name = stripsuffix(name)
293     }
294     mpx:execute(format("input %s ;", name))
295     if dump then
296       mpx:execute("dump ;")
297       info("format %s made and dumped for %s in %0.3f seconds", mem_name, name, os.clock()-
298         t)
299     else
300       info("%s read in %0.3f seconds", name, os.clock()-t)
301     end
302     return mpx
303   end
304
305   function luamplib.load(name)
306     local mem_name = replacesuffix(name, "mem")
307     local mpx = mplib.new {
308       ini_version = false,
309       mem_name = mem_name,
310       find_file = luamplib.finder
311     }
312     if not mpx and type(luamplib.make) == "function" then
313       -- when i have time i'll locate the format and dump
314       mpx = luamplib.make(name, mem_name)
315     end
316     if mpx then
317       info("using format %s", mem_name, false)
318       return mpx, nil
319     else
320       return nil, { status = 99, error = "out of memory or invalid format" }
321     end
322   end
323 end
```

```

322
323 else
324

```

These are the versions called with sufficiently recent mplib.

```

325 local preamble = [[
326   boolean mplib ; mplib := true ;
327   let dump = endinput ;
328   let normalfontsize = fontsize;
329   input %s ;
330 ]]
331
332 luamplib.make = luamplib.make or function()
333 end
334
335 function luamplib.load(name,verbatim)
336   local mpx = mplib.new {
337     ini_version = true,
338     find_file = luamplib.finder,

```

Provides numbersystem option since v2.4. Default value "scaled" can be changed by declaring \mplibnumbersystem{double}. See <https://github.com/lualatex/luamplib/issues/21>.

```

339     math_mode = luamplib.numbersystem,
340     random_seed = randomseed,
341   }

```

Append our own preamble to the preamble above.

```

342 local preamble = preamble .. (verbatim and "" or luamplib.mplibcodepreamble)
343 if luamplib.texttextlabel then
344   preamble = preamble .. (verbatim and "" or luamplib.texttextlabelpreamble)
345 end
346 local result
347 if not mpx then
348   result = { status = 99, error = "out of memory"}
349 else
350   result = mpx:execute(format(preamble, replacesuffix(name,"mp")))
351 end
352 luamplib.reporterror(result)
353 return mpx, result
354 end
355
356 end
357
358 local currentformat = "plain"
359
360 local function setformat (name) --- used in .sty
361   currentformat = name
362 end
363 luamplib.setformat = setformat

```

```

364
365
366 luamplib.reporterror = function (result)
367   if not result then
368     err("no result object returned")
369   else
370     local t, e, l = result.term, result.error, result.log
371     local log = stringgsub(t or l or "no-term", "%s+", "\n")
372     luamplib.lastlog = luamplib.lastlog .. "\n " .. (l or t or "no-log")
373     if result.status > 0 then
374       warn("%s", log)
375       if result.status > 1 then
376         err("%s", e or "see above messages")
377       end
378     end
379     return log
380   end
381 end
382
383 local function process_indeed (mpx, data, indeed)
384   local converted, result = false, {}
385   if mpx and data then
386     result = mpx:execute(data)
387     local log = luamplib.reporterror(result)
388     if indeed and log then
389       if luamplib.showlog then
390         info("%s", luamplib.lastlog)
391         luamplib.resetlastlog()
392       elseif result.fig then
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.
393         if stringfind(log, "\n>>") then info("%s", log) end
394         converted = luamplib.convert(result)
395       else
396         info("%s", log)
397         warn("No figure output. Maybe no beginfig/endfig")
398       end
399     end
400   else
401     err("Mem file unloadable. Maybe generated with a different version of mplib?")
402   end
403   return converted, result
404 end
405
v2.9 has introduced the concept of 'code inherit'
406 luamplib.codeinherit = false
407 local mplibinstances = {}
408 local process = function (data, indeed, verbatim)

```

workaround issue #70

```
409 if not stringfind(data, begname.."beginfig%s*%([%+%-]s)*%d[%.%d%s]*%") then
410   data = data .. "beginfig(-1);endfig;"
411 end
412 local standalone, firstpass = not luamplib.codeinherit, not indeed
413 local currfmt = currentformat .. (luamplib.numbersystem or "scaled")
414 currfmt = firstpass and currfmt or (currfmt.."2")
415 local mpx = mplibinstances[currfmt]
416 if standalone or not mpx then
417   randomseed = firstpass and math.random(65535) or randomseed
418   mpx = luamplib.load(currentformat,verbatim)
419   mplibinstances[currfmt] = mpx
420 end
421 return process_indeed(mpx, data, indeed)
422 end
423 luamplib.process = process
424
425 local function getobjects(result,figure,f)
426   return figure:objects()
427 end
428
429 local function convert(result, flusher)
430   luamplib.flush(result, flusher)
431   return true -- done
432 end
433 luamplib.convert = convert
434
435 local function pdf_startfigure(n,llx,lly,urx,ury)
```

The following line has been slightly modified by Kim.

```
436   texsprintf(format("\\mplibstarttoPDF{%f}{%f}{%f}{%f}", llx, lly, urx, ury))
437 end
438
439 local function pdf_stopfigure()
440   texsprintf("\\mplibstoptoPDF")
441 end
442
```

tex.tprint and catcode regime -2, as sometimes # gets doubled in the argument of pdfliteral. — modified by Kim

```
443 local function pdf_literalcode(fmt,...) -- table
444   textprint({"\\mplibtoPDF{"},{-2,format(fmt,...)},{""}})
445 end
446 luamplib.pdf_literalcode = pdf_literalcode
447
448 local function pdf_textfigure(font,size,text,width,height,depth)
```

The following three lines have been modified by Kim.

```
449 -- if text == "" then text = "\0" end -- char(0) has gone
450 text = text:gsub(".",function(c)
451   return format("\\hbox{\\char%i}",string.byte(c)) -- kerning happens in metapost
```

```

452 end)
453 texsprintf(format("\mplibtexttext{%s}{%f}{%s}{%s}{%f}", font,size,text,0,-( 7200/ 7227)/65536*depth))
454 end
455 luamplib.pdf_textfigure = pdf_textfigure
456
457 local bend_tolerance = 131/65536
458
459 local rx, sx, sy, ry, tx, ty, divider = 1, 0, 0, 1, 0, 0, 1
460
461 local function pen_characteristics(object)
462   local t = mplib.pen_info(object)
463   rx, ry, sx, sy, tx, ty = t.rx, t.ry, t.sx, t.sy, t.tx, t.ty
464   divider = sx*sy - rx*ry
465   return not (sx==1 and rx==0 and ry==0 and sy==1 and tx==0 and ty==0), t.width
466 end
467
468 local function concat(px, py) -- no tx, ty here
469   return (sy*px-ry*py)/divider,(sx*py-rx*px)/divider
470 end
471
472 local function curved(ith,pth)
473   local d = pth.left_x - ith.right_x
474   if abs(ith.right_x - ith.x_coord - d) <= bend_tolerance and abs(pth.x_coord - pth.left_x - d) <= bend_tolerance then
475     d = pth.left_y - ith.right_y
476     if abs(ith.right_y - ith.y_coord - d) <= bend_tolerance and abs(pth.y_coord - pth.left_y - d) <= bend_tolerance then
477       return false
478     end
479   end
480   return true
481 end
482
483 local function flushnormalpath(path,open)
484   local pth, ith
485   for i=1,#path do
486     pth = path[i]
487     if not ith then
488       pdf_literalcode("%f %f m",pth.x_coord,pth.y_coord)
489     elseif curved(ith,pth) then
490       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)
491     else
492       pdf_literalcode("%f %f l",pth.x_coord,pth.y_coord)
493     end
494     ith = pth
495   end
496   if not open then
497     local one = path[1]
498     if curved(pth,one) then
499       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)
500     else
501       pdf_literalcode("%f %f l",one.x_coord,one.y_coord)

```

```

502   end
503 elseif #path == 1 then
504   -- special case .. draw point
505   local one = path[1]
506   pdf_literalcode("%f %f l",one.x_coord,one.y_coord)
507 end
508 return t
509 end
510
511 local function flushconcatpath(path,open)
512 pdf_literalcode("%f %f %f %f %f %f cm", sx, rx, ry, sy, tx ,ty)
513 local pth, ith
514 for i=1,#path do
515   pth = path[i]
516   if not ith then
517     pdf_literalcode("%f %f m",concat(pth.x_coord,pth.y_coord))
518   elseif curved(ith,pth) then
519     local a, b = concat(ith.right_x,ith.right_y)
520     local c, d = concat(pth.left_x,pth.left_y)
521     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(pth.x_coord, pth.y_coord))
522   else
523     pdf_literalcode("%f %f l",concat(pth.x_coord, pth.y_coord))
524   end
525   ith = pth
526 end
527 if not open then
528   local one = path[1]
529   if curved(pth,one) then
530     local a, b = concat(pth.right_x,pth.right_y)
531     local c, d = concat(one.left_x,one.left_y)
532     pdf_literalcode("%f %f %f %f %f %f c",a,b,c,d,concat(one.x_coord, one.y_coord))
533   else
534     pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
535   end
536 elseif #path == 1 then
537   -- special case .. draw point
538   local one = path[1]
539   pdf_literalcode("%f %f l",concat(one.x_coord,one.y_coord))
540 end
541 return t
542 end
543

```

Below code has been contributed by Dohyun Kim. It implements btex / etex functions.

v2.1: `texttext()` is now available, which is equivalent to `TEX()` macro from `TEX.mp`.

`TEX()` is synonym of `texttext()` unless `TEX.mp` is loaded.

v2.2: Transparency and Shading

v2.3: `\everymplib`, `\everyendmplib`, and allows naked `TEX` commands.

```

544 local further_split_keys = {
545   ["MPLibTEXboxID"] = true,

```



```

546 ["sh_color_a"] = true,
547 ["sh_color_b"] = true,
548 }
549
550 local function script2table(s)
551   local t = {}
552   for _,i in ipairs(stringexplode(s,"\13+")) do
553     local k,v = stringmatch(i,"(.-)=(.*)") -- v may contain = or empty.
554     if k and v and k ~= "" then
555       if further_split_keys[k] then
556         t[k] = stringexplode(v,":")
557       else
558         t[k] = v
559       end
560     end
561   end
562   return t
563 end
564
565 local mplibcodepreamble = [[
566 vardef rawtexttext (expr t) =
567   if unknown TEXBOX_:
568     image( special "MPlibmkTEXbox="&t;
569     addto currentpicture doublepath unitsquare; )
570   else:
571     TEXBOX_ := TEXBOX_ + 1;
572     if known TEXBOX_wd_[TEXBOX_]:
573       image ( addto currentpicture doublepath unitsquare
574       xscaled TEXBOX_wd_[TEXBOX_]
575       yscaled (TEXBOX_ht_[TEXBOX_] + TEXBOX_dp_[TEXBOX_])
576       shifted (0, -TEXBOX_dp_[TEXBOX_])
577       withprescript "MPlibTEXboxID=" &
578       decimal TEXBOX_ & ":" &
579       decimal TEXBOX_wd_[TEXBOX_] & ":" &
580       decimal(TEXBOX_ht_[TEXBOX_]+TEXBOX_dp_[TEXBOX_]); )
581     else:
582       image( special "MPlibTEXError=1"; )
583   fi
584 fi
585 enddef;
586 if known context_mlib:
587   defaultfont := "cmtt10";
588   let infont = normalinfont;
589   let fontsize = normalfontsize;
590   vardef thelabel@#(expr p,z) =
591     if string p :
592       thelabel@#(p infont defaultfont scaled defaultscale,z)
593     else :
594       p shifted (z + labeloffset*mfun_laboff@# -
595       (mfun_labxf@#*lrcorner p + mfun_labyf@#*ulcorner p +

```

```

596      (1-mfun_labxf@#-mfun_labyf@#)*llcorner p))
597  fi
598  endif;
599  def graphicstext primary filename =
600    if (readfrom filename = EOF):
601      errmessage "Please prepare '"&filename&"' in advance with"&
602      " 'pstoedit -ssp -dt -f mpost yourfile.ps "&filename&"';
603    fi
604    closefrom filename;
605    def data_mpy_file = filename endif;
606    mfun_do_graphic_text (filename)
607  endif;
608  if unknown TEXBOX_: def mfun_do_graphic_text text t = endif; fi
609 else:
610   vardef texttext@# (text t) = rawtexttext (t) endif;
611 fi
612 def externalfigure primary filename =
613   draw rawtexttext("\includegraphics{"& filename &}")
614 endif;
615 def TEX = texttext endif;
616 def specialVerbatimTeX (text t) = special "MPLibVerbTeX="&t; endif;
617 def normalVerbatimTeX (text t) = special "PostMPLibVerbTeX="&t; endif;
618 let VerbatimTeX = specialVerbatimTeX;
619 extra_beginfig := extra_beginfig & " let VerbatimTeX = normalVerbatimTeX;" ;
620 extra_endfig := extra_endfig & " let VerbatimTeX = specialVerbatimTeX;" ;
621 ]]
622 luamplib.mplibcodepreamble = mplibcodepreamble
623
624 local texttextlabelpreamble = [[
625 primarydef s infont f = rawtexttext(s) endif;
626 def fontsize expr f =
627   begingroup
628   save size,pic; numeric size; picture pic;
629   pic := rawtexttext("\hskip\pdffontsize\font");
630   size := xpart urcorner pic - xpart llcorner pic;
631   if size = 0: 10pt else: size fi
632   endgroup
633 endif;
634 ]]
635 luamplib.texttextlabelpreamble = texttextlabelpreamble
636
637 local TeX_code_t = {}
638 local texboxnum = { 2047 }
639
640 local function domakeTEXboxes (data)
641   local num = texboxnum[1]
642   texboxnum[2] = num
643   local global = luamplib.globaltexttext and "\global" or ""
644   if data and data.fig then
645     local figures = data.fig

```

```

646   for f=1, #figures do
647       TeX_code_t[f] = nil
648       local figure = figures[f]
649       local objects = getobjects(data,figure,f)
650       if objects then
651           for o=1,#objects do
652               local object = objects[o]
653               local prescript = object.prescript
654               prescript = prescript and script2table(prescript)
655               local str = prescript and prescript.MPlibmkTEXbox
656               if str then
657                   num = num + 1
658                   texprescript(format("%s\\setbox%i\\hbox{%s}", global, num, str))
659               end
660               local texcode = prescript and prescript.MPlibVerbTeX
661               if texcode and texcode ~= "" then
662                   TeX_code_t[f] = texcode
663               end
664           end
665       end
666   end
667 end
668 if luamplib.globaltexttext then
669     texboxnum[1] = num
670 end
671 end
672
673 local function protect_tex_text_common (data)
674     local everymplib = texgettoks('everymplibtoks') or ''
675     local everyendmplib = texgettoks('everyendmplibtoks') or ''
676     data = format("\n%s\n%s\n%s",everymplib, data, everyendmplib)
677     data = data:gsub("\r","\n")
678
679     data = data:gsub("\n[\^n]-\n", function(str)
680         return str:gsub("([bem])tex"..endname,"%1"..esctex)
681     end)
682
683     data = data:gsub(btex_etex, function(str)
684         return format("rawtexttext(\\"%s\\"",protecttexcontents(str))
685     end)
686     data = data:gsub(verbatimtex_etex, function(str)
687         return format("VerbatimTeX(\\"%s\\"",protecttexcontents(str))
688     end)
689
690     return data
691 end
692

```

verbatimtex ... etex before beginfig() is not ignored, but the TeX code inbetween is inserted before the mplib box.

```

693 local function protecttexttextVerbatim(data)
694   data = protect_tex_text_common(data)
695
696   data = data:gsub("\^[^\n]-\\"", function(str) -- restore string btex .. etex
697     return str:gsub("[bem]".escstex, "%1tex")
698   end)
699
700   local _,result = process(data, false)
701   domakeTEXboxes(result)
702   return data
703 end
704
705 luamplib.protecttexttextVerbatim = protecttexttextVerbatim
706
707 luamplib.mpxcolors = {}
708
709 local function protecttexttext(data)
710   data = protect_tex_text_common(data)
711
712   data = data:gsub("\^[^\n]-\\"", function(str)
713     str = str:gsub("[bem]".escstex, "%1tex")
714       :gsub("%%", escpcnt)
715       :gsub("{", esclbr)
716       :gsub("}", escrbr)
717       :gsub("#", eschash)
718     return format("\detokenize{%s}",str)
719   end)
720
721   data = data:gsub("%%.-\n", "")
722
723   local grouplevel = tex.currentgrouplevel
724   luamplib.mpxcolors[grouplevel] = {}
725   data = data:gsub("\mpcolor".endname.."(.-){(.-)}", function(opt,str)
726     local cnt = #luamplib.mpxcolors[grouplevel] + 1
727     luamplib.mpxcolors[grouplevel][cnt] = format(
728       "\expandafter\mplibcolor\csname mpxcolor%i:%i\endcsname%s{%s}",
729       grouplevel,cnt,opt,str)
730     return format("\csname mpxcolor%i:%i\endcsname",grouplevel,cnt)
731   end)
732
733   Next line to address bug #55
734   data = data:gsub("([\^'\\\\])#", "%1##")
735   texpstr(data)
736 end
737
738 luamplib.protecttexttext = protecttexttext
739
740 local function makeTEXboxes (data)

```

```

741 data = data:gsub("##", "#")
742       :gsub(escpcnt, "%%")
743       :gsub(esclbrace, "{")
744       :gsub(escrbrace, "}")
745       :gsub(eschash, "#")
746 local _, result = process(data, false)
747 domakeTEXboxes(result)
748 return data
749 end
750
751 luamplib.makeTEXboxes = makeTEXboxes
752
753 local factor = 65536*(7227/7200)
754
755 local function processwithTEXboxes (data)
756   if not data then return end
757   local num = texboxnum[2]
758   local preamble = format("TEXBOX_:=%i;\n", num)
759   while true do
760     num = num + 1
761     local box = texgetbox(num)
762     if not box then break end
763     preamble = format(
764       "%sTEXBOX_wd_[%i]:=%f;\nTEXBOX_ht_[%i]:=%f;\nTEXBOX_dp_[%i]:=%f;\n",
765       preamble,
766       num, box.width /factor,
767       num, box.height/factor,
768       num, box.depth /factor)
769   end
770   process(preamble .. data, true)
771 end
772 luamplib.processwithTEXboxes = processwithTEXboxes
773
774 local pdfoutput = tonumber(texget("outputmode")) or tonumber(texget("pdfoutput"))
775 local pdfmode = pdfoutput > 0
776
777 local function start_pdf_code()
778   if pdfmode then
779     pdf_literalcode("q")
780   else
781     texsprint("\special{pdf:bcontent}") -- dvipdfmx
782   end
783 end
784 local function stop_pdf_code()
785   if pdfmode then
786     pdf_literalcode("Q")
787   else
788     texsprint("\special{pdf:econtent}") -- dvipdfmx
789   end
790 end

```

```

791
792 local function putTEXboxes (object,prescript)
793   local box = prescript.MPLibTEXboxID
794   local n,tw,th = box[1],tonumber(box[2]),tonumber(box[3])
795   if n and tw and th then
796     local op = object.path
797     local first, second, fourth = op[1], op[2], op[4]
798     local tx, ty = first.x_coord, first.y_coord
799     local sx, rx, ry, sy = 1, 0, 0, 1
800     if tw ~= 0 then
801       sx = (second.x_coord - tx)/tw
802       rx = (second.y_coord - ty)/tw
803       if sx == 0 then sx = 0.00001 end
804     end
805     if th ~= 0 then
806       sy = (fourth.y_coord - ty)/th
807       ry = (fourth.x_coord - tx)/th
808       if sy == 0 then sy = 0.00001 end
809     end
810     start_pdf_code()
811     pdf_literalcode("%f %f %f %f %f %f cm",sx,rx,ry,sy,tx,ty)
812     texsprint(format("\mplibputtextbox{%i}",n))
813     stop_pdf_code()
814   end
815 end
816

```

### Transparency and Shading

```

817 local pdf_objs = {}
818 local token, getpageres, setpageres = newtoken or token
819 local pgf = { bye = "pgfutil@everybye", extgs = "pgf@sys@addpdfresource@extgs@plain" }
820
821 if pdfmode then -- repeat luaotfload-colors
822   getpageres = pdf.getpageresources or function() return pdf.pageresources end
823   setpageres = pdf.setpageresources or function(s) pdf.pageresources = s end
824 else
825   texsprint("\special{pdf:obj @MPLibTr<<>>}",
826             "\special{pdf:obj @MPLibSh<<>>}")
827 end
828
829 -- objstr <string> => obj <number>, new <boolean>
830 local function update_pdfobjs (os)
831   local on = pdf_objs[os]
832   if on then
833     return on,false
834   end
835   if pdfmode then
836     on = pdf.immediateobj(os)
837   else
838     on = pdf_objs.cnt or 0

```

```

839 pdf_objs.cnt = on + 1
840 end
841 pdf_objs[os] = on
842 return on,true
843 end
844
845 local transparency_modes = { [0] = "Normal",
846 "Normal",      "Multiply",    "Screen",      "Overlay",
847 "SoftLight",   "HardLight",   "ColorDodge", "ColorBurn",
848 "Darken",      "Lighten",     "Difference",  "Exclusion",
849 "Hue",         "Saturation",  "Color",      "Luminosity",
850 "Compatible",
851 }
852
853 local function update_tr_res(res,mode,opaq)
854 local os = format("<</BM /%s/ca %.3f/CA %.3f/AIS false>>",mode,opaq,opaq)
855 local on, new = update_pdfobjs(os)
856 if new then
857   if pdfmode then
858     res = format("%s/MPLibTr%i %i 0 R",res,on,on)
859   else
860     if pgf.loaded then
861       texpstr(format("\\csname %s\\endcsname{MPLibTr%i}s}", pgf.extgs, on, os))
862     else
863       texpstr(format("\\special{pdf:put @MPLibTr<</MPLibTr%i%s>>}",on,os))
864     end
865   end
866 end
867 return res,on
868 end
869
870 local function tr_pdf_pageresources(mode,opaq)
871 if token and pgf.bye and not pgf.loaded then
872   pgf.loaded = token.create(pgf.bye).cmdname == "assign_toks"
873   pgf.bye = pgf.loaded and pgf.bye
874 end
875 local res, on_on, off_on = "", nil, nil
876 res, off_on = update_tr_res(res, "Normal", 1)
877 res, on_on = update_tr_res(res, mode, opaq)
878 if pdfmode then
879   if res ~= "" then
880     if pgf.loaded then
881       texpstr(format("\\csname %s\\endcsname{%s}", pgf.extgs, res))
882     else
883       local tpr, n = getpageres() or "", 0
884       tpr, n = tpr:gsub("/ExtGState<<", "%1"..res)
885       if n == 0 then
886         tpr = format("%s/ExtGState<<%s>>", tpr, res)
887       end
888       setpageres(tpr)

```

```

889     end
890   end
891   else
892     if not pgf.loaded then
893       texsprint(format("\\special{pdf:put @resources<</ExtGState @MPLibTr>>}"))
894     end
895   end
896   return on_on, off_on
897 end
898
899 local shading_res
900
901 local function shading_initialize ()
902   shading_res = {}
903   if pdfmode and luatexbase.callbacktypes and luatexbase.callbacktypes.finish_pdffile then -- ltuatex
904     local shading_obj = pdf.reserveobj()
905     setpagers(format("%s/Shading %i 0 R", getpagers() or "", shading_obj))
906     luatexbase.add_to_callback("finish_pdffile", function()
907       pdf.immediateobj(shading_obj, format("<<%s>>", tableconcat(shading_res)))
908     end, "luamplib.finish_pdffile")
909     pdf_objs.finishpdf = true
910   end
911 end
912
913 local function sh_pdfpagersources(shtype, domain, colorspace, colora, colorb, coordinates)
914   if not shading_res then shading_initialize() end
915   local os = format("<</FunctionType 2/Domain [ %s ]/C0 [ %s ]/C1 [ %s ]/N 1>>",
916     domain, colora, colorb)
917   local funcobj = pdfmode and format("%i 0 R", update_pdfobjs(os)) or os
918   os = format("<</ShadingType %i/ColorSpace /%s/Function %s/Coords [ %s ]/Extend [ true true ]/AntiAlia
919     shtype, colorspace, funcobj, coordinates)
920   local on, new = update_pdfobjs(os)
921   if pdfmode then
922     if new then
923       local res = format("/MPLibSh%i %i 0 R", on, on)
924       if pdf_objs.finishpdf then
925         shading_res[#shading_res+1] = res
926       else
927         local pageres = getpagers() or ""
928         if not stringfind(pageres, "/Shading<<.*>>") then
929           pageres = pageres.."/Shading<<>>"
930         end
931         pageres = pageres:gsub("/Shading<<\"", "%1"..res)
932         setpagers(pageres)
933       end
934     end
935   else
936     if new then
937       texsprint(format("\\special{pdf:put @MPLibSh<</MPLibSh%i%s>>}", on, os))
938     end

```



```

939   texsprintf(format("\special{pdf:put @resources<</Shading @MPLibSh>>}"))
940   end
941   return on
942 end
943
944 local function color_normalize(ca,cb)
945   if #cb == 1 then
946     if #ca == 4 then
947       cb[1], cb[2], cb[3], cb[4] = 0, 0, 0, 1-cb[1]
948     else -- #ca = 3
949       cb[1], cb[2], cb[3] = cb[1], cb[1], cb[1]
950     end
951   elseif #cb == 3 then -- #ca == 4
952     cb[1], cb[2], cb[3], cb[4] = 1-cb[1], 1-cb[2], 1-cb[3], 0
953   end
954 end
955
956 local prev_override_color
957
958 local function do_preobj_color(object,prescript)
959   -- transparency
960   local opaq = prescript and prescript.tr_transparency
961   local tron_no, troff_no
962   if opaq then
963     local mode = prescript.tr_alternative or 1
964     mode = transparency_modes[tonumber(mode)]
965     tron_no, troff_no = tr_pdf_pageresources(mode,opaq)
966     pdf_literalcode("/MPLibTr%i gs",tron_no)
967   end
968   -- color
969   local override = prescript and prescript.MPLibOverrideColor
970   if override then
971     if pdfmode then
972       pdf_literalcode(override)
973       override = nil
974     else
975       texsprintf(format("\special{color push %s}",override))
976       prev_override_color = override
977     end
978   else
979     local cs = object.color
980     if cs and #cs > 0 then
981       pdf_literalcode(luamplib.colorconverter(cs))
982       prev_override_color = nil
983     elseif not pdfmode then
984       override = prev_override_color
985       if override then
986         texsprintf(format("\special{color push %s}",override))
987       end
988     end

```

```

989 end
990 -- shading
991 local sh_type = prescript and prescript.sh_type
992 if sh_type then
993     local domain = prescript.sh_domain
994     local centera = stringexplode(prescript.sh_center_a)
995     local centerb = stringexplode(prescript.sh_center_b)
996     for _,t in pairs({centera,centerb}) do
997         for i,v in ipairs(t) do
998             t[i] = format("%f",v)
999         end
1000     end
1001     centera = tableconcat(centera," ")
1002     centerb = tableconcat(centerb," ")
1003     local colora = prescript.sh_color_a or {0};
1004     local colorb = prescript.sh_color_b or {1};
1005     for _,t in pairs({colora,colorb}) do
1006         for i,v in ipairs(t) do
1007             t[i] = format("%.3f",v)
1008         end
1009     end
1010     if #colora > #colorb then
1011         color_normalize(colora,colorb)
1012     elseif #colorb > #colora then
1013         color_normalize(colorb,colora)
1014     end
1015     local colorspace
1016     if #colorb == 1 then colorspace = "DeviceGray"
1017     elseif #colorb == 3 then colorspace = "DeviceRGB"
1018     elseif #colorb == 4 then colorspace = "DeviceCMYK"
1019     else return troff_no,override
1020     end
1021     colora = tableconcat(colora, " ")
1022     colorb = tableconcat(colorb, " ")
1023     local shade_no
1024     if sh_type == "linear" then
1025         local coordinates = tableconcat({centera,centerb}," ")
1026         shade_no = sh_pdfpageresources(2,domain,colorspace,colora,colorb,coordinates)
1027     elseif sh_type == "circular" then
1028         local radiusa = format("%f",prescript.sh_radius_a)
1029         local radiusb = format("%f",prescript.sh_radius_b)
1030         local coordinates = tableconcat({centera,radiusa,centerb,radiusb}," ")
1031         shade_no = sh_pdfpageresources(3,domain,colorspace,colora,colorb,coordinates)
1032     end
1033     pdf_literalcode("q /Pattern cs")
1034     return troff_no,override,shade_no
1035 end
1036 return troff_no,override
1037 end
1038

```

```

1039 local function do_postobj_color(tr,over,sh)
1040   if sh then
1041     pdf_literalcode("W n /MPlibSh%s sh Q",sh)
1042   end
1043   if over then
1044     texpstr("\special{color pop}")
1045   end
1046   if tr then
1047     pdf_literalcode("/MPlibTr%i gs",tr)
1048   end
1049 end
1050

```

End of btex – etex and Transparency/Shading patch.

```

1051
1052 local function flush(result,flusher)
1053   if result then
1054     local figures = result.fig
1055     if figures then
1056       for f=1, #figures do
1057         info("flushing figure %s",f)
1058         local figure = figures[f]
1059         local objects = getobjects(result,figure,f)
1060         local fignum = tonumber(stringmatch(figure:filename(),"([%d]+)$") or figure:charcode() or 0)
1061         local miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1062         local bbox = figure:boundingbox()
1063         local llx, lly, urx, ury = bbox[1], bbox[2], bbox[3], bbox[4] -- faster than unpack
1064         if urx < llx then

```

luamplib silently ignores this invalid figure for those codes that do not contain `beginfig ... endfig`. (issue #70)

```

1065         -- invalid
1066         -- pdf_startfigure(fignum,0,0,0,0)
1067         -- pdf_stopfigure()
1068       else

```

Insert `verbatimtex` code before `mplib box`. And prepare for those codes that will be executed afterwards.

```

1069         if TeX_code_t[f] then
1070           texpstr(TeX_code_t[f])
1071         end
1072         local TeX_code_bot = {} -- PostVerbatimTeX
1073         pdf_startfigure(fignum,llx,lly,urx,ury)
1074         start_pdf_code()
1075         if objects then
1076           local savedpath = nil
1077           local savedhtap = nil
1078           for o=1,#objects do
1079             local object      = objects[o]
1080             local objecttype  = object.type

```

Change from ConTeXt code: the following 7 lines are part of the `btex...etex` patch. Again, colors are processed at this stage. Also, we collect TeX codes that will be executed after flushing.

```

1081         local prescript      = object.prescript
1082         prescript = prescript and script2table(prescript) -- prescript is now a table
1083         local tr_opaq,cr_over,shade_no = do_preobj_color(object,prescript)
1084         if prescript and prescript.MPlibTEXboxID then
1085             putTEXboxes(object,prescript)
1086         elseif prescript and prescript.PostMPlibVerbTeX then
1087             TeX_code_bot[#TeX_code_bot+1] = prescript.PostMPlibVerbTeX
1088         elseif objecttype == "start_bounds" or objecttype == "stop_bounds" then
1089             -- skip
1090         elseif objecttype == "start_clip" then
1091             local evenodd = not object.istext and object.postscript == "evenodd"
1092             start_pdf_code()
1093             flushnormalpath(object.path,t,false)
1094             pdf_literalcode(evenodd and "W* n" or "W n")
1095         elseif objecttype == "stop_clip" then
1096             stop_pdf_code()
1097             miterlimit, linecap, linejoin, dashed = -1, -1, -1, false
1098         elseif objecttype == "special" then
1099             -- not supported
1100             if prescript and prescript.MPlibTEXError then
1101                 warn("texttext() anomaly. Try disabling \mplibtexttextlabel.")
1102             end
1103         elseif objecttype == "text" then
1104             local ot = object.transform -- 3,4,5,6,1,2
1105             start_pdf_code()
1106             pdf_literalcode("%f %f %f %f %f %f cm",ot[3],ot[4],ot[5],ot[6],ot[1],ot[2])
1107             pdf_textfigure(object.font,object.dsize,object.text,object.width,object.height,object.d)
1108             stop_pdf_code()
1109         else

```

Color stuffs are modified and moved to several lines above.

```

1110         local evenodd, collect, both = false, false, false
1111         local postscript = object.postscript
1112         if not object.istext then
1113             if postscript == "evenodd" then
1114                 evenodd = true
1115             elseif postscript == "collect" then
1116                 collect = true
1117             elseif postscript == "both" then
1118                 both = true
1119             elseif postscript == "eoboth" then
1120                 evenodd = true
1121                 both = true
1122             end
1123         end
1124         if collect then
1125             if not savedpath then

```

```

1126         savedpath = { object.path or false }
1127         savedhtap = { object.htap or false }
1128     else
1129         savedpath[#savedpath+1] = object.path or false
1130         savedhtap[#savedhtap+1] = object.htap or false
1131     end
1132 else
1133     local ml = object.miterlimit
1134     if ml and ml ~= miterlimit then
1135         miterlimit = ml
1136         pdf_literalcode("%f M",ml)
1137     end
1138     local lj = object.linejoin
1139     if lj and lj ~= linejoin then
1140         linejoin = lj
1141         pdf_literalcode("%i j",lj)
1142     end
1143     local lc = object.linecap
1144     if lc and lc ~= linecap then
1145         linecap = lc
1146         pdf_literalcode("%i J",lc)
1147     end
1148     local dl = object.dash
1149     if dl then
1150         local d = format("[%s] %i d",tableconcat(dl.dashes or {}, " "),dl.offset)
1151         if d ~= dashed then
1152             dashed = d
1153             pdf_literalcode(dashed)
1154         end
1155     elseif dashed then
1156         pdf_literalcode("[ ] 0 d")
1157         dashed = false
1158     end
1159     local path = object.path
1160     local transformed, penwidth = false, 1
1161     local open = path and path[1].left_type and path[#path].right_type
1162     local pen = object.pen
1163     if pen then
1164         if pen.type == 'elliptical' then
1165             transformed, penwidth = pen_characteristics(object) -- boolean, value
1166             pdf_literalcode("%f w",penwidth)
1167             if objecttype == 'fill' then
1168                 objecttype = 'both'
1169             end
1170         else -- calculated by mplib itself
1171             objecttype = 'fill'
1172         end
1173     end
1174     if transformed then
1175         start_pdf_code()

```

```

1176         end
1177     if path then
1178     if savedpath then
1179         for i=1,#savedpath do
1180             local path = savedpath[i]
1181             if transformed then
1182                 flushconcatpath(path,open)
1183             else
1184                 flushnormalpath(path,open)
1185             end
1186         end
1187         savedpath = nil
1188     end
1189     if transformed then
1190         flushconcatpath(path,open)
1191     else
1192         flushnormalpath(path,open)
1193     end

```

Change from ConTeXt code: color stuff

```

1194         if not shade_no then ----- conflict with shading
1195         if objecttype == "fill" then
1196             pdf_literalcode(evenodd and "h f*" or "h f")
1197         elseif objecttype == "outline" then
1198             if both then
1199                 pdf_literalcode(evenodd and "h B*" or "h B")
1200             else
1201                 pdf_literalcode(open and "S" or "h S")
1202             end
1203         elseif objecttype == "both" then
1204             pdf_literalcode(evenodd and "h B*" or "h B")
1205         end
1206     end
1207 end
1208 if transformed then
1209     stop_pdf_code()
1210 end
1211 local path = object.htap
1212 if path then
1213     if transformed then
1214         start_pdf_code()
1215     end
1216     if savedhtap then
1217         for i=1,#savedhtap do
1218             local path = savedhtap[i]
1219             if transformed then
1220                 flushconcatpath(path,open)
1221             else
1222                 flushnormalpath(path,open)
1223             end

```

```

1224         end
1225         savedhtap = nil
1226         evenodd = true
1227     end
1228     if transformed then
1229         flushconcatpath(path,open)
1230     else
1231         flushnormalpath(path,open)
1232     end
1233     if objecttype == "fill" then
1234         pdf_literalcode(evenodd and "h f*" or "h f")
1235     elseif objecttype == "outline" then
1236         pdf_literalcode(open and "S" or "h S")
1237     elseif objecttype == "both" then
1238         pdf_literalcode(evenodd and "h B*" or "h B")
1239     end
1240     if transformed then
1241         stop_pdf_code()
1242     end
1243     end
1244     end
1245     end

```

Added to ConT<sub>E</sub>Xt code: color stuff. And execute verbatimex codes.

```

1246         do_postobj_color(tr_opaq,cr_over,shade_no)
1247     end
1248     end
1249     stop_pdf_code()
1250     pdf_stopfigure()
1251     if #TeX_code_bot > 0 then
1252         texpstr(TeX_code_bot)
1253     end
1254     end
1255     end
1256     end
1257     end
1258 end
1259 luamplib.flush = flush
1260
1261 local function colorconverter(cr)
1262     local n = #cr
1263     if n == 4 then
1264         local c, m, y, k = cr[1], cr[2], cr[3], cr[4]
1265         return format("%.3f %.3f %.3f %.3f k %.3f %.3f %.3f %.3f K",c,m,y,k,c,m,y,k), "0 g 0 G"
1266     elseif n == 3 then
1267         local r, g, b = cr[1], cr[2], cr[3]
1268         return format("%.3f %.3f %.3f rg %.3f %.3f %.3f RG",r,g,b,r,g,b), "0 g 0 G"
1269     else
1270         local s = cr[1]
1271         return format("%.3f g %.3f G",s,s), "0 g 0 G"

```

```

1272 end
1273 end
1274 luamplib.colorconverter = colorconverter

```

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

```

1275  $\langle$ *package $\rangle$ 

```

First we need to load some packages.

```

1276 \bgroup\expandafter\expandafter\expandafter\egroup
1277 \expandafter\ifx\csname selectfont\endcsname\relax
1278 \input ltluatex
1279 \else
1280 \NeedsTeXFormat{LaTeX2e}
1281 \ProvidesPackage{luamplib}
1282 [2018/01/04 v2.12.2 mplib package for LuaTeX]
1283 \ifx\newluafunction\@undefined
1284 \input ltluatex
1285 \fi
1286 \fi

```

Loading of lua code.

```

1287 \directlua{require("luamplib")}

```

Support older formats

```

1288 \ifx\scantextokens\undefined
1289 \let\scantextokens\luatexscantextokens
1290 \fi
1291 \ifx\pdfoutput\undefined
1292 \let\pdfoutput\outputmode
1293 \protected\def\pdfliteral{\pdfextension literal}
1294 \fi

```

Set the format for metapost.

```

1295 \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.

```

1296 \ifnum\pdfoutput>0
1297 \let\mplibtoPDF\pdfliteral
1298 \else
1299 \def\mplibtoPDF#1{\special{pdf:literal direct #1}}
1300 \ifcsname PackageWarning\endcsname
1301 \PackageWarning{luamplib}{take dvipdfmx path, no support for other dvi tools currently.}
1302 \else
1303 \write128{}
1304 \write128{luamplib Warning: take dvipdfmx path, no support for other dvi tools currently.}
1305 \write128{}
1306 \fi
1307 \fi
1308 \def\mplibsetupcatcodes{%
1309 %catcode'\={12 %catcode'\}=12

```



```

1310 \catcode'\#=12 \catcode'\^=12 \catcode'\~=12 \catcode'\_ =12
1311 \catcode'\&=12 \catcode'\$=12 \catcode'\%=12 \catcode'\^^M=12 \endlinechar=10
1312 }

```

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

```

1313 \def\mplibputtextbox#1{\vbox to 0pt{\vss\hbox to 0pt{\raise\dp#1\copy#1\hss}}}
1314 \newcount\mplibstartlineno
1315 \def\mplibpostmpcatcodes{%
1316 \catcode'\{=12 \catcode'\}=12 \catcode'\#=12 \catcode'\%=12 }
1317 \def\mplibreplacelinebr{%
1318 \begingroup \mplibpostmpcatcodes \mplibdoreplacelinebr}
1319 \begingroup\lccode'\~='^^^M \lowercase{\endgroup
1320 \def\mplibdoreplacelinebr#1^^J{\endgroup\scantextokens{{}#1~}}

```

The Plain-specific stuff.

```

1321 \bgroup\expandafter\expandafter\expandafter\egroup
1322 \expandafter\ifx\csname selectfont\endcsname\relax
1323 \def\mplibreplacelinescs{%
1324 \begingroup \mplibpostmpcatcodes \mplibdoreplacelinescs}
1325 \begingroup\lccode'\~='^^^M \lowercase{\endgroup
1326 \def\mplibdoreplacelinescs#1^^J{\endgroup\scantextokens{\relax#1~}}
1327 \def\mplibcode{%
1328 \mplibstartlineno\inputlineno
1329 \begingroup
1330 \begingroup
1331 \mplibsetupcatcodes
1332 \mplibdocode
1333 }
1334 \long\def\mplibdocode#1\endmplibcode{%
1335 \endgroup
1336 \ifdefined\mplibverbatimYes
1337 \directlua{luamplib.tempdata\the\currentgrouplevel=luamplib.protecttexttextVerbatim(===[\detokenize
1338 \directlua{luamplib.processwithTEXboxes(luamplib.tempdata\the\currentgrouplevel)}]}%
1339 \else
1340 \edef\mplibtemp{\directlua{luamplib.protecttexttext(===[\unexpanded{#1}]===)}}}%
1341 \directlua{ tex.sprint(luamplib.mpxcolors[\the\currentgrouplevel]) }%
1342 \directlua{luamplib.tempdata\the\currentgrouplevel=luamplib.makeTEXboxes(===[\mplibtemp]===)}%
1343 \directlua{luamplib.processwithTEXboxes(luamplib.tempdata\the\currentgrouplevel)}%
1344 \fi
1345 \endgroup
1346 \ifnum\mplibstartlineno<\inputlineno\expandafter\mplibreplacelinescs\fi
1347 }
1348 \else

```

The  $\LaTeX$ -specific parts: a new environment.

```

1349 \newenvironment{mplibcode}{%
1350 \global\mplibstartlineno\inputlineno
1351 \toks@{\}\ltxdomplibcode
1352 }{}
1353 \def\ltxdomplibcode{%
1354 \begingroup

```

```

1355 \mplibsetupcatcodes
1356 \ltxdomplibcodeindeed
1357 }
1358 \def\mplib@mplibcode{mplibcode}
1359 \long\def\ltxdomplibcodeindeed#1\end#2{%
1360 \endgroup
1361 \toks@\expandafter{\the\toks@#1}%
1362 \def\mplibtemp@a{#2}\ifx\mplib@mplibcode\mplibtemp@a
1363 \ifdefined\mplibverbatimYes
1364 \directlua{luampLib.tempdata\the\currentgrouplevel=luampLib.protecttexttextVerbatim(===[\the\toks@]===)}%
1365 \directlua{luampLib.processwithTEXboxes(luampLib.tempdata\the\currentgrouplevel)}%
1366 \else
1367 \edef\mplibtemp{\directlua{luampLib.protecttexttext(===[\the\toks@]===)}}%
1368 \directlua{ tex.sprint(luampLib.mpxcolors[\the\currentgrouplevel]) }%
1369 \directlua{luampLib.tempdata\the\currentgrouplevel=luampLib.makeTEXboxes(===[\mplibtemp]===)}%
1370 \directlua{luampLib.processwithTEXboxes(luampLib.tempdata\the\currentgrouplevel)}%
1371 \fi
1372 \end{mplibcode}%
1373 \ifnum\mplibstartlineno<\inputlineno
1374 \expandafter\expandafter\expandafter\mplibreplacenewlinebr
1375 \fi
1376 \else
1377 \toks@\expandafter{\the\toks@\end{#2}}\expandafter\ltxdomplibcode
1378 \fi
1379 }
1380 \fi
1381 \def\mplibverbatim#1{%
1382 \begingroup
1383 \def\mplibtempa{#1}\def\mplibtempb{enable}%
1384 \expandafter\endgroup
1385 \ifx\mplibtempa\mplibtempb
1386 \let\mplibverbatimYes\relax
1387 \else
1388 \let\mplibverbatimYes\undefined
1389 \fi
1390 }

\everymplib & \everyendmplib: macros redefining \everymplibtoks & \everyendmplibtoks
respectively
1391 \newtoks\everymplibtoks
1392 \newtoks\everyendmplibtoks
1393 \protected\def\everymplib{%
1394 \mplibstartlineno\inputlineno
1395 \begingroup
1396 \mplibsetupcatcodes
1397 \mplibdoeverymplib
1398 }
1399 \long\def\mplibdoeverymplib#1{%
1400 \endgroup
1401 \everymplibtoks{#1}%

```

```

1402 \ifnum\mplibstartlineno<\inputlineno\expandafter\mplibreplacelinebr\fi
1403 }
1404 \protected\def\everyendmplib{%
1405 \mplibstartlineno\inputlineno
1406 \begingroup
1407 \mplibsetupcatcodes
1408 \mplibdoeveryendmplib
1409 }
1410 \long\def\mplibdoeveryendmplib#1{%
1411 \endgroup
1412 \everyendmplibtoks{#1}%
1413 \ifnum\mplibstartlineno<\inputlineno\expandafter\mplibreplacelinebr\fi
1414 }
1415 \def\mpdim#1{ \begingroup \the\dimexpr #1\relax\space \endgroup } % gmp.sty

```

Support color/xcolor packages. User interface is: `\mpcolor{teal}` or `\mpcolor[HTML]{008080}`, for example.

```

1416 \def\mplibcolor#1{%
1417 \def\set@color{\edef#1{1 withprescript "MPLibOverrideColor=\current@color"}}%
1418 \color
1419 }
1420 \def\mplibnumbersystem#1{\directlua{luamplib.numbersystem = "#1"}}
1421 \def\mplibmakenocache#1{\mplibdomakenocache #1, *, }
1422 \def\mplibdomakenocache#1,{%
1423 \ifx\empty#1\empty
1424 \expandafter\mplibdomakenocache
1425 \else
1426 \ifx*#1\else
1427 \directlua{luamplib.noneedtoreplace["#1.mp"]=true}%
1428 \expandafter\expandafter\expandafter\mplibdomakenocache
1429 \fi
1430 \fi
1431 }
1432 \def\mplibcancelnocache#1{\mplibdocancelnocache #1, *, }
1433 \def\mplibdocancelnocache#1,{%
1434 \ifx\empty#1\empty
1435 \expandafter\mplibdocancelnocache
1436 \else
1437 \ifx*#1\else
1438 \directlua{luamplib.noneedtoreplace["#1.mp"]=false}%
1439 \expandafter\expandafter\expandafter\mplibdocancelnocache
1440 \fi
1441 \fi
1442 }
1443 \def\mplibcachedir#1{\directlua{luamplib.getcachedir("\unexpanded{#1}")}}
1444 \def\mplibtexttextlabel#1{%
1445 \begingroup
1446 \def\tempa{enable}\def\tempb{#1}%
1447 \ifx\tempa\tempb
1448 \directlua{luamplib.texttextlabel = true}%

```

```

1449 \else
1450 \directlua{luamplib.texttextlabel = false}%
1451 \fi
1452 \endgroup
1453 }
1454 \def\mplibcodeinherit#1{%
1455 \begingroup
1456 \def\tempa{enable}\def\tempb{#1}%
1457 \ifx\tempa\tempb
1458 \directlua{luamplib.codeinherit = true}%
1459 \else
1460 \directlua{luamplib.codeinherit = false}%
1461 \fi
1462 \endgroup
1463 }
1464 \def\mplibglobaltexttext#1{%
1465 \begingroup
1466 \def\tempa{enable}\def\tempb{#1}%
1467 \ifx\tempa\tempb
1468 \directlua{luamplib.globaltexttext = true}%
1469 \else
1470 \directlua{luamplib.globaltexttext = false}%
1471 \fi
1472 \endgroup
1473 }

    We use a dedicated scratchbox.
1474 \ifx\mplibscratchbox\undefined \newbox\mplibscratchbox \fi

    We encapsulate the literals.
1475 \def\mplibstarttoPDF#1#2#3#4{%
1476 \hbox\bgroup
1477 \xdef\MPllx{#1}\xdef\MPlly{#2}%
1478 \xdef\MPurx{#3}\xdef\MPury{#4}%
1479 \xdef\MPwidth{\the\dimexpr#3bp-#1bp\relax}%
1480 \xdef\MPheight{\the\dimexpr#4bp-#2bp\relax}%
1481 \parskip0pt%
1482 \leftskip0pt%
1483 \parindent0pt%
1484 \everypar{}%
1485 \setbox\mplibscratchbox\vbox\bgroup
1486 \noindent
1487 }

1488 \def\mplibstoptoPDF{%
1489 \egroup %
1490 \setbox\mplibscratchbox\hbox %
1491 {\hskip-\MPllx bp%
1492 \raise-\MPlly bp%
1493 \box\mplibscratchbox}%
1494 \setbox\mplibscratchbox\vbox to \MPheight
1495 {\vfill

```

```

1496 \hsize\MPwidth
1497 \wd\mplibscratchbox0pt%
1498 \ht\mplibscratchbox0pt%
1499 \dp\mplibscratchbox0pt%
1500 \box\mplibscratchbox}%
1501 \wd\mplibscratchbox\MPwidth
1502 \ht\mplibscratchbox\MPheight
1503 \box\mplibscratchbox
1504 \egroup
1505 }

```

Text items have a special handler.

```

1506 \def\mplibtexttext#1#2#3#4#5{%
1507 \begingroup
1508 \setbox\mplibscratchbox\hbox
1509 {\font\temp=#1 at #2bp%
1510 \temp
1511 #3}%
1512 \setbox\mplibscratchbox\hbox
1513 {\hskip#4 bp%
1514 \raise#5 bp%
1515 \box\mplibscratchbox}%
1516 \wd\mplibscratchbox0pt%
1517 \ht\mplibscratchbox0pt%
1518 \dp\mplibscratchbox0pt%
1519 \box\mplibscratchbox
1520 \endgroup
1521 }

```

input luamplib.cfg when it exists

```

1522 \openin0=luamplib.cfg
1523 \ifeof0 \else
1524 \closein0
1525 \input luamplib.cfg
1526 \fi

```

That's all folks!

```

1527 </package>

```

# 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 programs, and that you know who you can do these things. To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

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

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

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

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

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

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

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

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

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

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

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

- You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
- You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole to 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. This is not the intent of this section to claim rights or contest your rights to control the distribution of derivative or collective works based on the Program.

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

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

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

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

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

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

8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit 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 rights 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.

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

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

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

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

### NO WARRANTY

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

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

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

signature of Ty Coon, 4 April 1989  
Ty Coon, President of Vice

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