Metapost Luafication

Taco Hoekwater

September 21, 2021

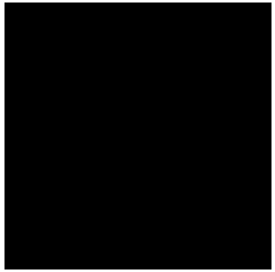
This talk explains how to call lua from Metafun code.

Let's do a very simple example first ...

```
\startluacode
function MP.doit(a)
    mp.print("unitsquare scaled " .. a)
end
\stopluacode

\startMPcode
   fill lua.MP.doit(100);
\stopMPcode
```

MetaPost Luafication



- ☐ you define a lua function in the `MP` table: `MP.doit()`
- ☐ that function uses one of the `mp.xxx()` helpers to produce output
- ☐ the METAPOST macro `lua` converts its arguments into a Lua call
- ☐ and converts the output back into METAPOST code

Now, let me try to explain what actually happens. First, on the Lua side ...

- The compiled MPLib library contains an extension that adds a new METAPOST primitive operation with the name `runscript`.
- This new primitive accepts a string as input and (is expected to) produce a string as output.
- The output of `runscript` is then internally converted back into METAPOST code.
- The script code to be run is set up during the creation of the METAPOST library instance.

MetaPost Luafication

Of course CONTEXT uses Lua for this extension, so it contains the following code:

```
local mpx, terminal = new_instance {
    ...
    run_script   = metapost.runscript,
    ...
}
```

MetaPost Luafication

`metapost.runscript` is a Lua function that uses `loadstring` to convert the input string into Lua code, and it returns an internal buffer as the result of the call to `runscript`.

That buffer is itself filled by the `mp.xxx()` helper functions.

MetaPost Luafication

On the METAPOST side, it would not be very nice if you had to code your Lua like this:

```
\startMPcode
   runscript ("MP.doit(" & decimal 100 & ")");
\stopMPcode
```

So, that is where the METAPOST `lua` macros comes in handy.

This is an METAPOST `vardef` that converts its suffixes and arguments into a Lua string for you, and then calls `runscript` internally.

That's it, really.

The Metafun manual has a list of those `mp.xxx()` helper functions.

Or you can look at `mlib-mpf` yourself (there are mkiv and lmtx versions).

When in doubt, you can always use `mp.print("string")`

Just make sure that the `"string"` is valid METAPOST.

Handy to know:

- ☐ The Metafun manual contains (much) more information than this talk
- ☐ METAPOST's `lua` is defined in `mp-luas` (again in mkiv and lmtx versions)
- ☐ Your Lua functions can do just about anything because the result is processed immediately.
- ☐ And there is a tracker you can turn on: `metapost.lua`