The vomit Class

Eito YONEYAMA

2025/09/02 v1.0

Abstract

This class replaces characters with the emoji 😫. It is mainly a demonstration of hooking into LuaTeX callbacks.

Contents

1 Introduction

The vomit class intercepts input lines with Lua callbacks and replaces every non-command character with the emoji 😫. It requires LuaLaTeX and the bxcoloremoji package.

2 Usage

Basic usage:

```
\documentclass{vomit}
\begin{document}
Hello, world!
\end{document}
```

To temporarily disable vomit mode, use `\stopvomit`. To re-enable it, use `\startvomit`. After calling `\startvomit`, all non-command characters are replaced by the vomit emoji 😫.

3 Implementation

```
1 ⟨*class⟩
2 %%
3 %% This is file ‘vomit.cls’,
4 %% generated with the docstrip utility.
```

```
%%
%% The original source files were:
%%
%% vomit.dtx  (with options: ‘class’)
%%
%% IMPORTANT NOTICE:
%%
%% For the copyright see the source file.
%%
%% Any modified versions of this file must be renamed
%% with new filenames distinct from vomit.cls.
%%
%% For distribution of the original source see the terms
%% for copying and modification in the file vomit.dtx.
%%
%% This generated file may be distributed as long as the
%% original source files, as listed above, are part of the
%% same distribution. (The sources need not necessarily be
%% in the same archive or directory.)
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{vomit}[2025/09/02 v1.0 Universal Emoji Replacer Class]

\LoadClass{article}
\RequirePackage{bxcoloremoji}
\RequirePackage{luacode}

\begin{luacode*}
local function utf8_len(s)
  if type(s) ~= "string" then return 0 end
  if unicode and unicode.utf8 and unicode.utf8.len then
    local ok, v = pcall(unicode.utf8.len, s)
    if ok and v then return v end
  end
  if utf8 and utf8.len then
    local ok, v = pcall(utf8.len, s)
    if ok and v then return v end
  end
  local n = 0
  for _ in s:gmatch("[%z\1-\127\194-\244][\128-\191]*") do n = n + 1 end
  return n
end

local function utf8_bytepos(s, n)
  if n < 1 then return nil end
  if unicode and unicode.utf8 and unicode.utf8.offset then
    local ok, pos = pcall(unicode.utf8.offset, s, n)
    if ok then return pos end
  end
  if utf8 and utf8.offset then
    local ok, pos = pcall(utf8.offset, s, n)
    if ok then return pos end
  end
  local i = 0
  for pos, _ in s:gmatch("()([%z\1-\127\194-\244][\128-\191]*)") do
```

```
    i = i + 1
    if i == n then return pos end
  end
  return nil
end

local function utf8_sub_chars(s, i, j)
  local len = utf8_len(s)
  if i < 1 then i = 1 end
  if j == nil or j > len then j = len end
  if i > j then return "" end
  local b1 = utf8_bytepos(s, i)
  local b2next = utf8_bytepos(s, j+1)
  local b2 = (b2next and (b2next - 1)) or #s
  if not b1 then return "" end
  return string.sub(s, b1, b2)
end

local function trim(s)
  return (s:gsub("^%s+", ""):gsub("%s+$", ""))
end

local function vomitify_argument(text)
  local len = utf8_len(text)
  local result = {}

  for i = 1, len do
    local ch = utf8_sub_chars(text, i, i)
    if ch:match("^[%w%s%p]$") and not ch:match("^[{}\\]$") then
      table.insert(result, "\\coloremojicode{1F92E}")
    else
      table.insert(result, ch)
    end
  end

  return table.concat(result)
end

local function process_command_arguments(line)
  local result = line:gsub("(\\%w+)({[^}]*})", function(cmd, arg)
    local cmd_name = cmd:match("\\(.+)")
    if cmd_name == "dummy" or cmd_name == "section" or cmd_name == "title" or
       cmd_name == "subsection" or cmd_name == "subsubsection" then
      local inner = arg:match("{(.*)}")
      if inner then
        local converted_inner = vomitify_argument(inner)
        return cmd .. "{" .. converted_inner .. "}"
      end
    end
    return cmd .. arg
  end)

  return result
end
```

```
113
114 local function vomitify_text(line)
115   if type(line) ~= "string" then
116     return line
117   end
118
119   if line:find("\\end{document}", 1, true) then
120     return line
121   end
122
123   local t = trim(line)
124
125   if t:match("^%%") then
126     return line
127   end
128
129   if t == "" then
130     return line
131   end
132
133   if t:match("^\\%w+{") then
134     return process_command_arguments(line)
135   end
136
137   if t:match("^\\") then
138     return line
139   end
140
141   local len = utf8_len(line)
142   local i = 1
143   local res = {}
144
145   while i <= len do
146     local ch = utf8_sub_chars(line, i, i)
147
148     if ch == "\\" then
149       local cmd = ch
150       local j = i + 1
151       while j <= len do
152         local nch = utf8_sub_chars(line, j, j)
153         if nch:match("^[A-Za-z]$") then
154           cmd = cmd .. nch
155           j = j + 1
156         else
157           break
158         end
159       end
160       if cmd == "\\" and j <= len then
161         local next_ch = utf8_sub_chars(line, j, j)
162         if not next_ch:match("^[A-Za-z]$") then
163           cmd = cmd .. next_ch
164           j = j + 1
165         end
166       end
```

```
      table.insert(res, cmd)
      i = j
    elseif ch == "{" or ch == "}" or ch == "&" or ch == "$" or ch:match("^%s$") then
      table.insert(res, ch)
      i = i + 1
    else
      table.insert(res, "\\coloremojicode{1F92E}")
      i = i + 1
    end
  end

  return table.concat(res)
end

function enable_vomit_mode()
  if luatexbase and luatexbase.add_to_callback then
    luatexbase.add_to_callback("process_input_buffer", vomitify_text, "vomitify_all_text")
  else
    callback.register("process_input_buffer", vomitify_text)
  end
end

function disable_vomit_mode()
  if luatexbase and luatexbase.remove_from_callback then
    luatexbase.remove_from_callback("process_input_buffer", "vomitify_all_text")
  else
    callback.register("process_input_buffer", nil)
  end
end
\end{luacode*}

\AtBeginDocument{%
  \directlua{enable_vomit_mode()}%
}

\newcommand{\stopvomit}{%
  \directlua{disable_vomit_mode()}%
}

\newcommand{\startvomit}{%
  \directlua{enable_vomit_mode()}%
}
⟨/class⟩
```