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

```
⟨*class⟩
\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

```

```lua
local function vomitify_text(line)
  if type(line) ~= "string" then
    return line
  end

  if line:find("\\end{document}", 1, true) then
    return line
  end

  local t = trim(line)
  if t:match("^\\") or t == "" or t:match("^%%") then
    return line
  end

  local len = utf8_len(line)
  local i = 1
  local res = {}

  while i <= len do
    local ch = utf8_sub_chars(line, i, i)

    if ch == "\\" then
      local cmd = ch
      local j = i + 1
      while j <= len do
        local nch = utf8_sub_chars(line, j, j)
        if nch:match("^[A-Za-z]$") then
          cmd = cmd .. nch
          j = j + 1
        else
          break
        end
      end
      if cmd == "\\" and j <= len then
        local next_ch = utf8_sub_chars(line, j, j)
        if not next_ch:match("^[A-Za-z]$") then
          cmd = cmd .. next_ch
          j = j + 1
        end
      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, "\\coloremoji{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⟩
```