# TSNET スクリプト通信


TSC 編集委員会

ISSN 1884-2798

# 目次



---

表紙写真:　練習船 日本丸

撮影: jscripter

日時: 2010 年 11 月 21 日

場所: 広島市宇品１万トンバース

メモ: さて、現実の日本丸はどこに向かうのか。練習船の日本丸と同様に優美に爽やかに輝きたいものである。

# 巻頭言

jscripter

TSNETスクリプト通信第3巻第3号、通算第11号が出る。その経過報告から。

クリスマスプレゼント刊行をアナウンスしてから、ほどなくムムリクさんからRubyTkスクリプトの投稿をいただいて喜んだ。そして、海鳥さんに「よしおさんとロボ太」掲載の了解をいただいた。海鳥さんが@niftyの黙示録サイトをちょうど閉鎖された後で連絡を入れたので、お手間を取らせて全データをパッケージにしていただくことになった。大変ありがとうございました。自分自身はと言えば、Twitterのプロファイルを検索するスクリプトを試していたのをなんとか完成させて記事を書いた。そして昨日、Yささんからarray.awkの投稿があり、最後によかったねと思った。そうして今、巻頭言を書いている。

みなさん、年末のお忙しい中、ご協力ありがとうございました。

(投稿: 2010年12月23日)

# 点取り虫ゲーム　Ruby/Tk 版

ムムリク

■点取り虫ゲームについて


テントリムシゲームのトップ画面

点取り虫ゲームは、月刊アスキー 1981 年 11 月号の TBN コーナーに掲載された N-BASIC で書かれたゲームプログラムです。すでに掲載号を所持していないので詳細は不明ですが、1981/8/9 Takagi Hitoshi （高木　均）さんによって作られました。

10x10 の升目に 1 から 5 の数字が割り振られていて、プレイヤーはそれぞれの角からゲームをはじめます。はじめは角にある数字が得点となり、次に進む場所はその数字分だけ縦・横・斜めに離れた場所になります。一度取った場所はもう取れません。取った点数が次に移動できる場所を決定します。そうして順に移動しつつ得点し、動けなくなったらそのプレイヤーのゲームは終了です。最終的にすべてのプレイヤーが移動できなくなった時の得点で勝者が決まります。


テントリムシゲーム画面

ルールはごく簡単ですが、次第に移動できる場所が限られてくるのに、移動できる場所を決定するのは取った点数によるので、なかなか難しいところもあります。

今でもエミュレータ上でプレイすることがありますが、今回このゲームを Ruby/Tk で書いてみました。

## ■遊び方

実行には、Ruby（1.9.2 または 1.8.7）、Tcl（おおむね 8.5） が必要です。Ruby のインストールパッケージなどによっては Ruby/Tk が入っていない場合があります。その場合には、入っているものを利用するか、Ruby/Tk Kit をインストールするなどしてください。（arton さん作成の ActiveScriptRuby や 1.9.x のパッケージには基本的に含まれています。RubyInstaller には含まれていないようです。Tcl は ActiveTcl8.5 をインストールしてください）

コンソールから、

```
>ruby tentorimushi.rb
```

とすればゲームのはじまりです。


**点取り虫ゲーム　Ruby/Tk 版**

起動時のプレイヤーは人間 1 人とコンピュータという設定になっています。コンピュータは常に右下の Red を担当します。プレイヤーの色で移動できる場所が示されるので、移動したい場所をクリックします。過去に移動した場所は灰色に変わり、以降その場所への移動はできません。

[設定]ボタンを押すと、プレイヤーの人数とコンピュータを入れるかどうかの設定ができます。プレイヤーの人数は最大で 4 人です。コンピュータを入れる場合には、人間は 3 人までです。


**点取り虫ゲーム　Ruby/Tk 版 設定画面**

ゲームが終了したところで、[New Game]ボタンを押すと、プレイヤーの設定はそのままで新しい盤面でゲームをはじめることができます。また、ゲームの途中で[設定]ボタンを押して人数を変更すると、その人数設定で新たに盤面が作られてゲームがはじまります。

■余談

コンピュータの思考はごく単純にしかしていないため、賢いということは一切ありません。とはいえ、いくら先読みをしても、プレイヤーの人数が増えてくると、自分の移動先を十分に予測することはなかなか困難です。先読みよりも運に左右される面もあり、そのあたりが面白さかもしれません。あるいは、思考アルゴリズムをきちんと作りこんで無敵のコンピュータを作りあげるというのも、楽しいかもしれません。

*Ruby/Tk-Kit ( http://www.dumbo.ai.kyutech.ac.jp/nagai/RubyTk/?Ruby%2FTk-Kit )

■スクリプト

---

```
# -*- coding: sjis -*-
require 'tk'
CELLSIZE = 30

class Tentorimushi
  def initialize(x=10)
    @cells = Array.new
    @width = x - 1
    @height = x - 1
  end
  attr_reader :width, :height

  def add(buttons)
    @cells << buttons
  end

  def where_move(player)
    (@cells[player.x])[player.y].text = '*' if (@cells[player.x])[player.y].text
!= 'E'
    (@cells[player.x])[player.y].background = player.color
    player.can_move.each do |x, y|
      (@cells[x])[y].background = player.color
    end
  end

  def geometry(x, y)
    (@cells[x])[y]
  end
```

```
  def move?(nowx, nowy, movex, movey)
    if nowx + movex >= 0 && nowx + movex < @width + 1
      if nowy + movey >= 0 && nowy + movey < @height + 1
        if /[1-5]/ =~ (@cells[nowx + movex])[nowy + movey].text
          [nowx + movex, nowy + movey]
        else
          false
        end
      else
        false
      end
    else
      false
    end
  end

end

class Player
  def initialize(color, sead = 'man')
    @sead = sead
    @point = 0
    @distance = 0
    @can_move = []
    @color = color
    @info_label = nil
    if color == 'yellow'
      set_yellow
    elsif color == 'red'
      set_red
```

```
    elsif color == 'skyblue'
      set_skyblue
    elsif color == 'green'
      set_green
    end
  end

  def set_yellow
    @x = 0
    @y = 0
  end
  def set_red
    @x = 9
    @y = 9
  end
  def set_skyblue
    @x = 9
    @y = 0
  end
  def set_green
    @x = 0
    @y = 9
  end
  attr_reader :sead, :color
  attr_accessor :point, :x, :y, :can_move, :distance, :info_label
end

class Players
  def initialize
    @now_play = []
    @list = []
```

```
  end

  def add(player)
    @list << player
  end

  attr_reader :list
  attr_accessor :now_play
end

def make_board(game)
  button = Array.new(game.width * game.height)
  t = Time.now
  srand(t.sec^t.usec)
  f = Array.new

  i = 0
  (0..game.height).each {|row|
    buttons = Array.new
    (0..game.width).each {|column|
              f = TkFrame.new(@f_center, 'width' => CELLSIZE, 'height' =>
CELLSIZE).grid(:row => row, :column => column)
      TkGrid.propagate(f, false)
           button[i] = TkButton.new(@f_center, 'text' => rand(5).to_i + 1,
'background' => 'white').grid('row' => row, 'column' => column, 'sticky' =>
'news')
       button[i].command proc { player_move(game, row, column) if correct_click?
(row, column) == true }
      buttons << button[i]
      i += 1
    }
```

```
    game.add(buttons)
  }
end

def move_area(distance)
  [[-distance, -distance], [0, -distance], [distance, -distance],
   [-distance, 0],                         [distance, 0],
   [-distance, distance], [0, distance],   [distance, distance]]
#
#      -y
#       |
#       |
# -x ---0--- +x
#       |
#       |
#      +y
end

def set_move_area(game, player)
  move_areas = move_area(player.distance)
  move_positions = []
  move_areas.each do |geom|
    position = game.move?(player.x, player.y, *geom)
    move_positions << position if position != false
  end
  player.can_move = move_positions
end

def correct_click?(x, y)
  return false if now_player == 'com'
  now_player.can_move.each do |cmx, cmy|
```

```
    if x == cmx && y == cmy
      return true
    end
  end
  return false
end

def player_move(game, x, y)
  if correct_click?(x, y) == true
    player = now_player
    player.can_move.each do |cmx,cmy|
      game.geometry(cmx, cmy).background = 'white' if x != cmx or y != cmy
    end
    player.distance = game.geometry(x, y).text.to_i
    player.point += player.distance
    player.info_label.text = player.point.to_s
    game.geometry(player.x, player.y).text = ''
    game.geometry(player.x, player.y).background = 'gray'
    player.x = x
    player.y = y
    game.geometry(x, y).text = '*'
    change_player
    player_alive?(game)
  else
    player_alive?(game)
  end
  player = now_player
  @l_top.text = "#{player.color} さんの番です。"
  game.where_move(player)
  if 'com' == player.sead && player.can_move.size > 0
    Thread.start{
```

```
      sleep 1
      com_move(game)
    }
  end
end

def player_alive?(game)
  player = before_move(game)
  if player.can_move.size == 0
    if @players.now_play.size > 1
      Thread.new{ info_finish(player) }
      game.geometry(player.x, player.y).text = 'E'
      @players.now_play.shift
      sleep 2
      player_alive?(game)
    else
      game.geometry(player.x, player.y).text = 'E'
      info_game_end
    end
  end
end

def com_move(game)
  player = now_player
  get_points = []
  player.can_move.each do |cmx, cmy|
    get_points << game.geometry(cmx, cmy).text
  end
  player.can_move.each do |cmx, cmy|
    if get_points.sort.reverse[0] == game.geometry(cmx, cmy).text
      @move_x = cmx
```

```
      @move_y = cmy
      break
    end
  end
  player_move(game, @move_x, @move_y)
end

def info_finish(player)
  w_info_player = TkToplevel.new(nil)
  w_info_player.title = "お知らせ"
  TkMessage.new(w_info_player, 'text' => "#{player.color} さんは、もう動けません。
", 'width' => 300).pack
          w_info_player.geometry("+#{TkWinfo.rootx(@root)     +     120}"     +
"+#{TkWinfo.rooty(@root) + 120}")
  w_info_player.focus
  w_info_player.grab
  sleep 2
  w_info_player.destroy
end

def info_game_end
  w_info_game_end = TkToplevel.new(nil)
  w_info_game_end.title = "勝負あり"
  m_game_end = TkMessage.new(w_info_game_end, 'width' => 300).pack
  win = @players.list.sort_by{|p| p.point}[@players.list.size - 1]
  m_game_end.text = "#{now_player.color} さんは、もう動けません。¥n#{win.color} さ
んの勝ちです。"
  b_info_game_end = TkButton.new(w_info_game_end, 'text' => 'OK', 'width' => 10,
'command' => proc {w_info_game_end.destroy; @l_top.text = "ゲーム終了";
@b_new_game.state('normal')}).pack
          w_info_game_end.geometry("+#{TkWinfo.rootx(@root)     +     120}"     +
```

```
"+#{TkWinfo.rooty(@root) + 120}")
  w_info_game_end.focus
  w_info_game_end.grab
end

def before_move(game)
  player = now_player
  set_move_area(game, player)
  player
end

def now_player
  @players.list[@players.now_play[0]]
end

def change_player
  rotate_player(@players) if @players.now_play.size > 1
end

def rotate_player(players)
  num = players.now_play.shift
  players.now_play.push(num)
end

def make_frame
     @f_top  =  TkFrame.new(@root,  'relief'  =>  'ridge',  'borderwidth'  =>
2).pack('side' => 'top', 'fill' => 'both')
     @f_left  =  TkFrame.new(@root,  'relief'  =>  'ridge',  'borderwidth'  =>
2).pack('side' => 'left', 'fill' => 'both')
  @f_center = TkFrame.new(@root).pack('side' => 'left')
     @f_right  =  TkFrame.new(@root,  'relief'  =>  'ridge',  'borderwidth'  =>
```

```
2).pack('side' => 'left', 'fill' => 'both')
end

def make_information
  top_info
  yellow_info
  skyblue_info
  red_info
  green_info
end

def top_info
   @b_new_game = TkButton.new(@f_top, 'text' => 'New Game', 'command' => proc
{ new_game }).pack('side' => 'left', 'padx' => 2, 'pady' => 2)
  @b_new_game.state('disabled')
  @b_config = TkButton.new(@f_top, 'text' => "設定", 'width' => 6, 'command' =>
proc { how_many_player? }).pack('side' => 'left', 'padx' => 2, 'pady' => 2)
   @l_top = TkLabel.new(@f_top, 'text' => '', 'width' => 30).pack('side' =>
'left', 'pady' => 2)
  @b_end_game = TkButton.new(@f_top, 'text' => "終了", 'width' => 6, 'command'
=> proc { @root.destroy }).pack('side' => 'right', 'padx' => 2, 'pady' => 2)
end

def yellow_info
    l_yellow = TkLabel.new(@f_left, 'text' => 'Yellow', 'background' =>
'yellow').pack('side' => 'top', 'fill' => 'x')
    @l_yellow_point = TkLabel.new(@f_left, 'width' => 8, 'text' => '0',
'background' => 'yellow').pack('side' => 'top', 'fill' => 'x')
end

def red_info
```

```
  @l_red_point = TkLabel.new(@f_right, 'width' => 8, 'text' => '0', 'background'
=> 'red').pack('side' => 'bottom', 'fill' => 'x')
      @l_red  =  TkLabel.new(@f_right,  'text'  =>  'Red',  'background'  =>
'red').pack('side' => 'bottom', 'fill' => 'x')
end

def skyblue_info
    @l_skyblue_point  =  TkLabel.new(@f_left,  'width'  =>  8,  'text'  =>  '0',
'background' => 'skyblue').pack('side' => 'bottom', 'fill' => 'x')
    l_skyblue  =  TkLabel.new(@f_left,  'text'  =>  'Skyblue',  'background'  =>
'skyblue').pack('side' => 'bottom', 'fill' => 'x')
end

def green_info
     l_green  =  TkLabel.new(@f_right,  'text'  =>  'Green',  'background'  =>
'green').pack('side' => 'top', 'fill' => 'x')
    @l_green_point  =  TkLabel.new(@f_right,  'width'  =>  8,  'text'  =>  '0',
'background' => 'green').pack('side' => 'top', 'fill' => 'x')
end

def how_many_player?
  @w_how_many = TkToplevel.new
  @w_how_many.title = "人数設定"
            @w_how_many.geometry("+#{TkWinfo.rootx(@root)      +      90}"      +
"+#{TkWinfo.rooty(@root) + 120}")
  if @players != nil
    use_com = TkVariable.new(@players.list[@players.list.size - 1].sead)
    if 'man' == use_com
      number_of_players = TkVariable.new(@players.list.size)
    else
      number_of_players = TkVariable.new(@players.list.size - 1)
```

```
    end
  else
    use_com = TkVariable.new('com')
    number_of_players = TkVariable.new(1)
  end
   f_use_com = TkFrame.new(@w_how_many, 'relief' => 'ridge', 'borderwidth' =>
2).pack('side' => 'top', 'fill' => 'x')
rb_com =  TkRadioButton.new(f_use_com, 'text' => "COM をいれる", 'value' =>
'com', 'variable' => use_com).pack('side' => 'left')
rb_man =  TkRadioButton.new(f_use_com, 'text' => "COM をいれない", 'value' =>
'man', 'variable' => use_com).pack('side' => 'left')

     f_number_of_players = TkFrame.new(@w_how_many, 'relief' => 'ridge',
'borderwidth' => 2).pack('side' => 'top', 'fill' => 'x')
    l_number_of_players = TkLabel.new(f_number_of_players, 'text' => "人の
数").pack('side' => 'top')
    TkRadioButton.new(f_number_of_players, 'text' => "１人", 'value' => 1,
'variable' => number_of_players).pack('side' => 'left')
    TkRadioButton.new(f_number_of_players, 'text' => "２人", 'value' => 2,
'variable' => number_of_players).pack('side' => 'left')
    TkRadioButton.new(f_number_of_players, 'text' => "３人", 'value' => 3,
'variable' => number_of_players).pack('side' => 'left')
  rb_number_of_players4 = TkRadioButton.new(f_number_of_players, 'text' => "４人
", 'value' => 4, 'variable' => number_of_players).pack('side' => 'left')
    rb_number_of_players4.state('disabled') if use_com.value == 'com'

  rb_com.bind 'Button', proc { rb_number_of_players4.state('disabled') }
  rb_man.bind 'Button', proc { rb_number_of_players4.state('normal') }

  f_button_ok = TkFrame.new(@w_how_many).pack('side' => 'left')
  f_button_cancel = TkFrame.new(@w_how_many).pack('side' => 'right')
```

```
  b_ok = TkButton.new(f_button_ok, 'text' => 'OK', 'width' => 10).pack('side' =>
'right', 'padx' => 20)
  b_ok.command = proc { set_player(use_com.value, number_of_players.numeric) }
   b_cancel = TkButton.new(f_button_cancel, 'text' => 'キャンセル', 'width' =>
10).pack('side' => 'left', 'padx' => 20)
  b_cancel.command = proc { @w_how_many.destroy }
  @w_how_many.focus
  @w_how_many.grab
end

def set_player(use_com, number_of_players)
  make_player(use_com, number_of_players)
  game_init
  game_start(@t)
end

def make_player(use_com = 'com', number_of_players = 1)
   colors = [['yellow', @l_yellow_point], ['green',@l_green_point], ['skyblue',
@l_skyblue_point], ['red', @l_red_point]]
  @players = Players.new
  number_of_players = 3 if 'com' == use_com && 4 == number_of_players
  (number_of_players).times do |num|
    p = Player.new( (colors[num])[0] )
    p.info_label = (colors[num])[1]
    @l_red.text = 'Red'
    @players.add p
    @players.now_play << num
  end
  com_player_set(@players, colors) if 'com' == use_com
  counter_reset(colors)
  @w_how_many.destroy if @w_how_many != nil
```

```
end

def com_player_set(players, colors)
  p = Player.new('red', 'com')
  p.info_label = (colors[3])[1]
  @l_red.text = '(com)'
  players.add p
  players.now_play << players.now_play.size
end

def counter_reset(colors)
  colors.each do |c|
    c[1].text = '0'
  end
end

def init_player(game, players)
  players.list.each_with_index do |p, i|
    p.point = game.geometry(p.x, p.y).text.to_i
    p.distance = p.point
    p.info_label.text = p.point.to_s
  end
end

def game_init
  @t = Tentorimushi.new
  make_board(@t)
  init_player(@t, @players)
end

def game_start(game)
```

```
  set_move_area(game, now_player)
  @l_top.text = "#{now_player.color} さんの番です。"
  game.where_move(now_player)
end

def new_game
  @b_new_game.state('disabled')
  make_player
  game_init
  game_start(@t)
end

@root = TkRoot.new
@root.title = "点取り虫ゲーム"
make_frame
make_information
new_game

Tk.mainloop
```

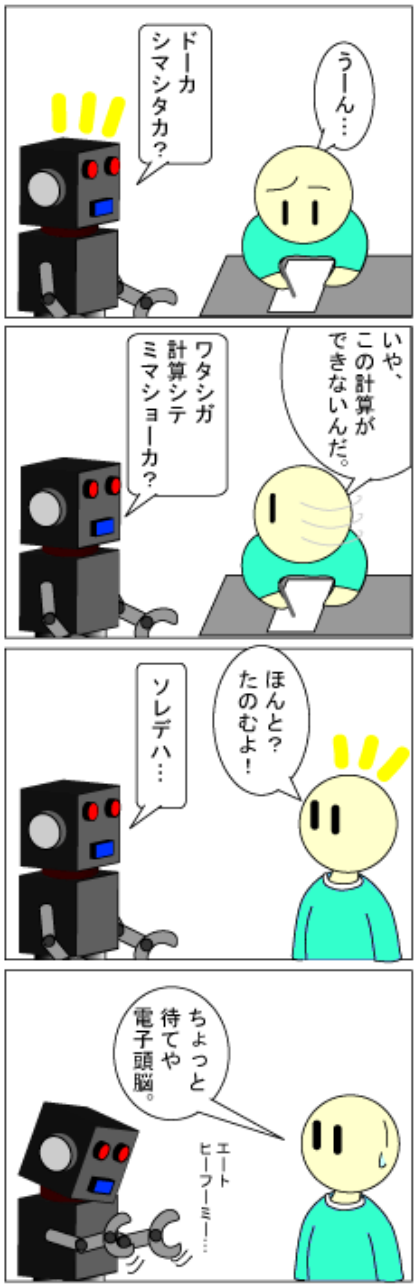
---

（投稿：2010年12月7日）

# よしおさんとロボ太

海鳥

「高性能計算機」

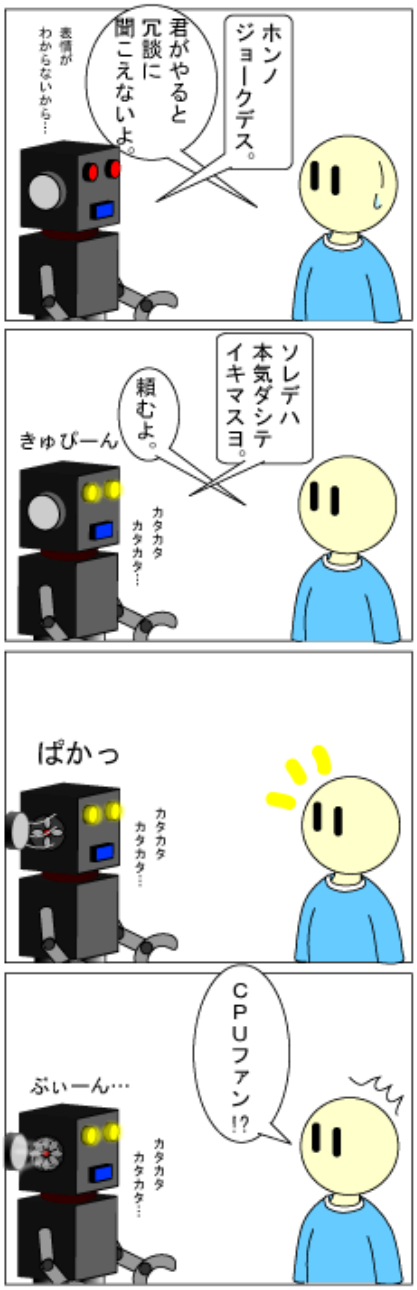



二進法？

# よしおさんとロボ太

海鳥

「続・高性能計算機」




耳の構造がついに明らかに。

# スクリプト工作入門
## - ツイッターを題材に -

jscripter

## 1. Twitter の時代

9 月くらいから本格的に Twitter を使い始めた。最初触ったのは 2007 年ぐらいだったが、あまり必然性を感じることができなかった。意味のあるところまで書くには 140 文字では足りないだろうと思った。写真やメモをケータイからモバイル時に投稿するという仕組みは興味深く思えたが、実際自分で試してみると中途半端で実用的でないと感じた。歩きながらなどの本当のモバイル時は、落ち着いて書く余裕がないので、メモも片言になってしまうからだ。手持ちのケータイの写真の画質に感心しなかったということも理由の一つだった。

今年、センサーの測定結果を自動的にツイートさせて、そのツイッターからデータを取得するアイデアを実行している話を読んで、なるほどと興味が回復した。様々な利用法がありうるものだと思ったからだ。今日で 1100 ツイートを達成。10 ツイート/日のペースでツイートしている。もちろんリツイートを含めてだけど、帰宅してからの少ない自由時間ではそれが精一杯であるが、逆に増えた個人的情報をどう取り扱うべきかというのはパーソナル情報処理において今後の課題だろう。フォローの選択と共に、ツイートはさらに絞るべきかもしれない。しかしながら、大量のツイートを積み重ねている先人たちを見ていると、ツイッターの時代が到来していると思う。

## 2. Twitter にメモ

インターネットで日々得る情報はツイッターに簡単にメモすることができる。

i. ツイッターで読んで気になるツイートはリツイートする。そうすれば、自分のプロファイルに取り込まれる。

ii. RSS/Atom などで配信されたり、Google などで検索した Web ページは、Firefox では Google ツールバーの共有を使って、ツイートすることができる。https のページは

Marginize を使えば可能である。

## 3. スクリプト工作入門

TSNET スクリプト通信 2.2(August 2009)に「Zed 入門 II - Twitter on Zed」という記事を書いて、実は 8 月にその続編を書こうとしているうちに、Basic 認証システムが OAuth 認証に変わり、OAuth を使ったアプリケーションを登録する必要が出てきた。 登録が面倒になって中止したのが前号までの経過であった。登録するには、次のページで「新しいアプリケーションを追加する」で設問に答えていく必要がある。適当に答えておくという手はあるのだが、作る前には答えにくい設問もある。作る前に何を作るか、あるいは作れるかを明確にイメージしていない場合もあり得るのだ。取り敢えず認証を通らないとプログラムをデバッグすることさえできないと思った。

Twitter / アプリケーション
http://twitter.com/oauth_clients

Twitter へのアプリケーション登録 |HAPPY*TRAP
http://www.happytrap.jp/blogs/2010/01/08/1694/

今調べてみると、登録内容は後から自由に変更できるのだそうだ。本稿のスクリプトもアプリケーション、tw_read.pl として一応登録したが、Twitter でログインせずに Twitter を読む方法を試してみよう。Twitter の利用のイメージを膨らませるために。

### 3.1. Twitter を読む

もちろん、Twitter のユーザーのプロファイルを読むためだけなら、Twitter を Web ブラウザで読めば良い。自作のプログラムで読む必要はない。しかし、プロファイルのツイートの保存は 3200 件までは保証されているようだが、それ以上は保証されないそうだ。ツイートをメモの集積として考えれば、過去のメモは自分で残して自在に扱える状態にしておく必要がある。それが最終的な目的だ。

テキスト処理の技術を使えば、対象となるテキストから必要な情報を抽出して自在に整形することができる。本稿では、Twitter のユーザープロファイルがどのような HTML で表現され、そこからどこまで必要なメモデータを抽出できるかを検討してみよう。

### 3.2. Twitter のユーザープロファイルの HTML

次に、http://twitter.com/jscripter などで得られる HTML のツイートとリツイートの部分を両方とも例示しよう(実在するものではない)。

#### 3.2.1 クラス: status-body

span タグの status-body のクラスを含む範囲にツイート関連のデータが含まれる。要素を色分けして表示しておくので参照されたい。ツイートは Twitter の用語では「status」なので、HTML のクラスは「status-body」となっている。status-body の主要部分は、status-content と meta entry-meta に分かれる。

---

```
          <li class="hentry u-jscripter status" id="status_16323742514814976"
>
    <span class="status-body">
    <span class="status-content">
                    <span class="entry-content">うーむ。そう言われてもね。これはテス
トなんだよね。 @<a class="tweet-url username" href="/mumeiken"
rel="nofollow">mumeiken</a> えーっ、それって確かなんだろうか</span>
          </span>
    <span class="meta entry-meta" data='{}'>
  <a class="entry-date" rel="bookmark"
href="http://twitter.com/jscripter/status/16323742514814976">
    <span class="published timestamp" data="{time:'Sun Dec 19 02:47:38 +0000
2010'}">約 1 時間前</span></a>
  <span>web から</span>
```

```
  <a href="http://twitter.com/memeiken/status/16280509831188480">mumeiken 宛
</a></span>

        <ul class="meta-data clearfix">
</ul>
  </span>
</li>
.....
<li  class="hentry u-mumeiken share status" id="status_29259955284"
>
      <span class="status-body">
              <span class="big-retweet-icon" title="Retweets from people you
follow appear in your timeline."></span>
                    <strong><a href="http://twitter.com/mumeiken" class="tweet-
url screen-name">mumeiken</a></strong>

                  <span class="entry-content">それには正確な時間が必要なんだよ
ね。QT @<a class="tweet-url username" href="/jscripter"
rel="nofollow">jscripter</a> そう言われてもね。</span>
            <span class="meta entry-meta" data='{}'>
  <a class="entry-date" rel="bookmark"
href="http://twitter.com/mumeiken/status/29244921729">
    <span class="published timestamp" data="{time:'Sun Oct 31 03:58:26 +0000
2010'}">8:58 PM Oct 30th</span></a>
  <span><a href="http://homepage99.notify.com/mumeiken/" rel="nofollow">無名犬
</a>から</span>

  </span>

        <span class="meta entry-meta retweet-meta">
          <span class="shared-content"><a href="/jscripter" class="screen-name
```

```
timestamp-title" data="&#123;time:&quot;Sun Oct 31 08:36:44 +0000
2010&quot;&#125;" title="2 日前">jscripter</a>と 2 人がリツイート</span>
        </span>

            <ul class="meta-data clearfix">
</ul>
    </span>
  </li >
```

---

### 3.2.2. クラス: status-content, entry-content

span タグの status-content クラスは span タグの entry-content クラスからなり、その名前のごとく、ツイートあるいはリツイートの内容である。

### 3.2.3. クラス: meta entry-meta, entry-date, published timestamp

span タグの「meta entry-meta」のクラスは、主に a タグの entry-date クラスと span タグの「published timestamp」クラスからなり、ツイートあるいはリツイートへの直接のリンクとツイート時のタイムスタンプが記録されている。

### 3.2.4. クラス: meta entry-meta retweet-meta, shared content, screen-name timestamp-title

「meta entry-meta retweet-meta」はリツイートに存在する span タグのクラスで、span タグの「shared content」クラスと a タグの「screen-name timestamp-title」クラスからなる。「shared content」には何人がリツイートしたかなどのリツイート関連の情報がまとめられている。「screen-name timestamp-title」クラスの screen-name は username と同じ。ユーザーのプロファイルのタイムラインにリンクが張られ、timestamp の生データが実体参照にエンコードされて、a タグの data 属性に置かれている。title 属性には、ツイートした時刻からの経過時間が「2 日前」などの言葉で表示されている。

## 3.3. Perl スクリプト: tw_read.pl

3.2 項にあるログインせずに得られる Twitter のプロファイルのタイムラインの HTML が解析の対象である。本稿ではあまり複雑なことは考えない。単にツイートあるいはリツイート毎の status-body を配列に格納して、再出力する。ツイートやリツイート内のデータを取得してデータベース的に取り扱うことは次稿で行う予定である。既に前項で内容を解析したので簡単にできそうだ。

スクリプトは UTF-8 の文字コードで書かれる必要がある。スクリプトに設定すれば、デフォルトの値だけで動作させることも可能だし、コマンドラインから必要なパラメータを取得して動作することもできる。もちろん、CGI として動作させ、フォームからパラメータを取得して動作することがもっとも望ましい。デスクトップ CGI で動作させることを前提としてセキュリティは考慮していないので、Web 上で動作させる場合にはご注意願いたい。

次にスクリプトをまず示す。コメントに理解するために必要なことは盛り込んだので大体はわかるはずである。後でスクリプトのポイントを少し説明をして終わろう。

[tw_read.pl]

```
#!/Perl5.8/bin/perl.exe
use LWP::Simple;
use CGI qw(:cgi);
use URI::Escape;

### 初期設定(デフォルト項目 3 箇所のみ設定してください)

# 本 CGI を置くパスの設定
my $cgi_path = "http://localhost/cgi-bin";# デフォルト

# 検索パターンの取得
my $pattern = ".";# デフォルト
if($ARGV[0]){
        $pattern = $ARGV[0];# コマンドライン第 1 引数から
```

```
}elsif(param('pattern')){
        $pattern = param('pattern');# CGI のパラメータ、pattern から
}

# username の取得
my $tw_user = "jscripter";# デフォルト
if($ARGV[1]){
        $tw_user = $ARGV[1];# コマンドライン第 2 引数から
}elsif(param('tw_user')){
        $tw_user = param('tw_user');# CGI のパラメータ、tw_user から
}

# ツイッタープロファイルの URL の取得
my $tw_url = "http://twitter.com/" . $tw_user;
# ページを移動する際、再帰的に本 CGI から URL を取得する場合に username を取得
if(param('tw_url')){
        $tw_url = param('tw_url');
        if($tw_url =~ /^http:¥/¥/twitter.com¥/([^¥?]+)$/){
             $tw_user = $1;# CGI パラメータを URL が持たない場合
        }elsif($tw_url =~ /^http:¥/¥/twitter.com¥/(.+?)¥?/){
             $tw_user = $1;# CGI パラメータを URL が持つ場合
        }
}

# 現在のページの取得
my $ppage = 0;
# ページを移動する際、再帰的に本 CGI から URL を取得する場合にページ数を取得
if($tw_url =~ /page=(¥d+)/){
        $ppage = $1;
}
```

```
# 次ページがない場合のフラグ(1: 次ページなし)
my $nopage = 0;

my $npage = 0;# 次ページ数
my $new_tw_url = "";# 次ページの URL
my $escaped_next_url = "";# 次ページの URL の値をエスケープしたもの

my $prevpage = 0;# 前ページ数
my $prev_tw_url = "";# 前ページの URL
my $escaped_prev_url = "";# 前ページの URL の値をエスケープしたもの

### CGI 最初の部分の出力

print <<HEADER;
Content-type: text/html; charset=UTF-8

<html>
<style type="text/css">
<!--
body { margin: 1em 2em; }
div,a { text-align: left; }
div.status-body { text-indent: 0em; font-family: monospace;
        width: 40em; line-height: 1.5em; background-color: lightyellow;
        border-width: 10px 20px 10px 10px; }
div.entry-content { text-indent: 1em; background-color: lightblue; }
div.meta { text-indent: 1em; background-color: lightyellow; }
div.shared-content { text-indent: 1em; background-color: lightyellow; }
-->
</style>
<body>
HEADER
```

```
###

# 読み込み URL の CGI 出力
print "<p>${tw_url}</p>¥n";

### ツイートデータの取得

# LWP::Simple の get メソッドに入力するための URL 変換
$tw_url =~ s/¥&amp;/¥&/g;

# LWP::Simple の get メソッドで URL に対応する HTML を得る
my $tw = get($tw_url);

# HTML 中のツイート部分を個別に配列に格納
# <li class="hentry...>から</li>あるいは</li >までの範囲を取得
my @status_bodys = ($tw =~ /<li¥s+class="hentry.+?>(.+?)<¥/li.*?>/sg);

# 次ページの URL を取得、現在のページ数の取得
if($tw =~ /<a href="(¥/${tw_user}¥?.+?)"/s){
        $next_tw_url = $1;# 次ページの URL
        if($next_tw_url =~ /page=(¥d+)/){
            $npage = $1;# 次ページ数
            $ppage = $npage - 1;# 現在のページ数
        }
}else{
        if($tw_url =~ /(¥/${tw_user}.+)/){
            $next_tw_url = $1;
        }
        $nopage = 1;# 次ページの URL が取得できない
}
```

```
if($ppage == 0){
        $ppage = 1;# 現在のページを取得できない場合など
}

###

# ページ数の CGI 出力
print "<center>" . $ppage . "</center>¥n";

print "<hr>";

### ツイートの CGI 出力
foreach (@status_bodys){
        if(/>.*?$pattern.*?</so){
            # 表示上、span タグは改行されないので、div タグに変換
            s/<span/<div/sg;
            s/<¥/span>/<¥/div>/sg;
            # @username や#hashtag の処理
            # "/username"や"/search?q=#hashtag"のハイパーリンクを補完する
            s/(<.*?)href="(¥/[^¥/]+?)"/${1}href="http:¥/¥/twitter.com$2"/sg;
            print $_;
        }
}
###

print "<hr>¥n";
print "<center>";

### ページの移動リンクの CGI 出力

# 戻るページへのリンク表示
```

```
if($ppage > 1){
        $prevpage = $ppage - 1;
        ($prev_tw_url = $next_tw_url) =~ s/page=¥d+/page=${prevpage}/;
        $escaped_prev_url = uri_escape("http://twitter.com${prev_tw_url}");
        print "<a href=¥"${cgi_path}/twitter_read.pl?pattern=$
{pattern}¥&tw_url=${escaped_prev_url}¥">${prevpage}ページへ戻る</a> <---> ";
}

# 進むページへのリンク表示
if($nopage){
        print "もうページはありません。¥n";# 最後のページの場合
}else{
        $escaped_next_url = uri_escape("http://twitter.com${next_tw_url}");
        print "<a href=¥"${cgi_path}/twitter_read.pl?pattern=$
{pattern}¥&tw_url=${escaped_next_url}¥">${npage}ページへ進む</a>¥n";
}

###

# CGIの最後の部分の出力
print "</center>¥n</body>¥n</html>¥n";
__END__
```

---

### 3.2.1. status-bodyの取得

```
-----------------------------------------------------------------
# HTML中のツイート部分を個別に配列に格納
# <li class="hentry...>から</li>あるいは</li >までの範囲を取得
my @status_bodys = ($tw =~ /<li¥s+class="hentry.+?>(.+?)<¥/li.*?>/sg);
-----------------------------------------------------------------
```

上記部分が、spanタグのstatus-bodyクラスの範囲を@status_bodyの配列に格納する部

分である。パターンマッチに正規表現修飾子の「s」を使っているところがポイントである。HTML 全体が$tw に格納されており、全体を１行と見做してパターンマッチを行う。修飾子の「g」も付いているから、マッチするすべてが配列に取り出される。

これでデータ処理としては大体勝負は付いたようなものである。が、ポイントはデータがこのような処理に都合の良い形式になっていたということである。対応関係のある<li class="hentry...>から</li>あるいは</li >までの範囲を取得すすればよい。最小マッチを使えば、データ範囲には</li>あるいは</li >が存在しないので、そのような芸当が可能となる。

ちなみに、li タグの終わりのタグに半角空白が含まれているとリツイートであり、半角空白が含まれなければツイートという規則性があった。その理由はまだわかっていない。

正規表現修飾子の「s」を今回のスクリプトは多用しているので、調べていただきたい。@status_body の配列要素の処理にも使っている。

### 3.2.2. CGI を呼び出すためのフォーム

次のような HTML を組み込んで動かすことになるでしょう。

---

```
<ul>
<!-- Twitter 検索用 CGI - tw_read.pl -->
<li><FORM action="cgi-bin/tw_read.pl" target="main">Twitter プロファイル検索<br>
        username: <INPUT type="text" name="tw_user" value="jscripter"
size="20"><br>
        検索パターン: <INPUT type="text" name="pattern" value="." size="20">
        <INPUT type="submit"><INPUT type="reset">
        </FORM>
<!-- -->
</ul>
```

### 3.2.3 スクリプトの動作と今後の展開

スクリプトはプロファイルのタイムラインの検索を意図している。しかしながら、現在のスクリプトは、タイムラインの最新のページから次第に過去のページに手動で遡りながら検索を繰り返すという仕様になっている。デザインも今一である。読者の方で関心をもたれた方がおられれば、是非改良したものを発表していただきたい。スクリプトは自由に使っていただいて結構である。

次に、デスクトップCGIとして動作しているところを示そう。

デスクトップCGIとして動作させたtw_read.pl

Twitter のユーザー名と検索パターンを入力するインターフェースの部分を示しておく。


**tw_read.pl の入力インターフェース**

さて、著者は余裕があれば、プロファイルタイムラインデータをデスクトップ上で更新しつつ格納しながら、検索も可能なスクリプトに改良していく予定である。そして、wemo や他のデスクトップ CGI と融合させるような方向を考えるかもしれない。

世界経済の異常事態にも関わらず、インターネット世界の進展はますます目覚ましいものがある。テレビはもうテレビとは言えなくなりつつある。テレビはコンピュータになり、ネットワークに接続された。もっとも、USTREAM を表示するにはまだ PC が必要なようではあるが・・・

すべてのモノがインターネットやコンピュータに接続されそうである。そして、Twitter と結び付けられるかもしれない。

（投稿: 2010 年 12 月 19 日）

# Array・おぼえていますか

written by Ｙさ

## 1. このゲームは

簡単に言うと記憶力を鍛えるゲームです。 ...って省略しすぎ(^^; 以前に古典的脳トレ(memory=いわゆるカード配置を高度な記憶力と勘で推理するゲーム)を投稿しましたが、今回は本格派(?)脳トレです。表示される図形を順番に記憶して、後から思い出して回答するゲームです。

二種類のルールが楽しめる"1粒で2度おいしい"スクリプトとなっております d(^^;

ルールの一つに、いわゆる"Nバック課題"を採用しています。非常にざっくり簡単に言うと「新しい事を記憶しつつ、N個前(例えば2つ前)に憶えたものを思い出す」課題です。ウィキペディアによるとオリジナルは1958年にキルヒナーによって紹介されたみたいです。最近になって脳トレブームで話題に上っているみたいですが... 詳しくはWIRED NEWSの記事(※)等を見てください。

※原文
http://www.wired.com/science/discoveries/news/2008/04/smart_software (英語)
http://wiredvision.jp/news/200805/2008050723.html (日本語版)

## 2. 動作環境は

例によって、最近のawkならOKだと思います。ちなみに動作確認は、

GNU Awk 3.1.8, mawk 1.3.3 MBCS R27

で(WinXPのDOS窓で)行なってます。

[注意]スクリプト内 function CLS() にもコメントしていますが、コマンドプロンプトのクリアをして図形表示しています。環境によっては "cls" ではダメなので、"clear" をお試しください。

## 3. 遊び方は

```
>gawk -f array.awk[Enter]
~~~~~~~~~~~~~~~~~~~~~~~~~
```

等で起動するとゲームスタートです。

1)最初にゲームの動作モードを選択してください。二種類のどちらかを'A'か'B'を入力して選びます。

'A' : [A]t once = 記憶した全てを一気に回答するモード
'B' : N-[B]ack task = N個前に表示されていたものを回答する、いわゆる"Nバック課題"モード
（入力は大文字小文字を区別しません。）

なお、

```
-v GAMEMODE="B"
~~~~~~~~~~~~~~~
```

といった感じでオプションを付けて起動しても、ゲームの動作モードを決定できます。

2)<動作モード'A'>

上下左右を表す図形が1つずつN個(2～16)まで表示されるので順番に向きを全て記憶してください。なお、上下左右を表す図形のパターンとしては3種類あり、それぞれ通常/反転表示があります。(つまり計6種類)

次にN個分の向きについて、文字をN個入力して[Enter]で回答してください。

向きに対応する文字：
上 = 'W' または 'I' または '8'
右 = 'D' または 'L' または '6'
下 = 'S' または 'K' または '5' または '2'
左 = 'A' または 'J' または '4'
（入力は大文字小文字を区別しません。）

同じ列の長さで3回出題を繰り返します。全てに正解し続けると、列の長さが増えます。(最大16個まで、それ以降は16)

N個全てを正解できなかった場合は、正解率を表示してゲーム終了となります。

3)<動作モード'B'>

上下左右を表す図形が1つずつ10+N*2個(N=2～10)表示されるので順番に向きを記憶していってください。

N+1個を表示するとN個前の向きを聞いてくるので、文字を1個入力して[Enter]で回答してください。
例)N=2の場合：3個目の図形を表示した際に1個目の図形の向きを聞いてくるので答える。

向きに対応する文字は動作モード'A'と同じです。

同じNの長さで6回出題を繰り返します。全てに正解し続けると、Nの長さが1つ増えます。(最大10個まで、それ以降は10)
10+N個全てを正解できなかった場合は、正解率を表示してゲーム終了となります。

## 4.開発環境など

ソースは、awk版として新規に作成したものです。ThinkPadのWinXP上のテキストエディタとコマンドプロンプトな環境でやってます。

前回の「古典的脳トレ」と違って、今回の題材は自分との戦いです。 ...ちなみに作者本人は 2-Back ですら満足にクリアできません(^^;

## 5. その他

本プログラムはフリーソフトです。
著作権は作者であるＹさにあります。
ただし転載、再配布、改造、消去は自由です。
(できましたら素晴らしい改造を加えた後に TSC 編集委員会宛に投稿してくださいませ)
また、このソフトを使用した事による損害が発生したとしても、損害に対しては一切の責任を負いかねます。
同様に、このソフトを使用して"脳"力がちっとも向上しなかったとしても一切の責任を負いかねます。

## 6. スクリプト

[array.awk]

```
##  Do You Remember Array?  written by Ｙさ

function CLS(){
  system("cls");  # clear screen --if "cls" does not work, try "clear"
}

function rnd(N){ return int(N * rand()); }  ##  乱数

BEGIN{
  srand();

  # ゲームモード決定
```

```
  gm=tolower(GAMEMODE);
  if(gm!="a" && gm!="b"){
    print "Which Mode? : [A]t once / N-[B]ack task";
    printf(" 'A'or'B'?> ");
    do{ gm=""; getline gm; gm=tolower(gm); }while(gm!="a" && gm!="b");
  }
  GAMEMODE=gm;

  # ゲーム準備(表示パターン設定、等)
  delete PatternTbl;
  MakePatternData(PatternTbl);
  Block[0]="  ";  Block[1]="**";
  Block[2]="%%";  Block[3]="  ";

  split("U,R,D,L", DirStr, ",");  DirStr[0]="?";
  Key2Dir["w"]=Key2Dir["i"]=Key2Dir["8"]=1;
  Key2Dir["d"]=Key2Dir["l"]=Key2Dir["6"]=2;
  Key2Dir["s"]=Key2Dir["k"]=Key2Dir["5"]=Key2Dir["2"]=3;
  Key2Dir["a"]=Key2Dir["j"]=Key2Dir["4"]=4;

  # ゲームメイン
  delete Score;  # 成績
  Stage = 0;
  do{
    ++Stage;
    Level = GetLevel(Stage,GAMEMODE);
    if(GAMEMODE=="a"){ DispTaskLimit=Level;    Offset=1; }
    if(GAMEMODE=="b"){ DispTaskLimit=Level+10; Offset=Level; }
  }while(RoundLoop(DispTaskLimit,Offset)==0);
  exit;
}
```

```
END{
  print "¥n¥n****** GAME OVER ******¥n¥n";

  for(st=1; st<=Stage; ++st){
    Level = GetLevel(st,GAMEMODE);
    if(GAMEMODE=="a") DispTaskLimit=Level;
    if(GAMEMODE=="b") DispTaskLimit=Level+10;
    DispResult(st,DispTaskLimit,((GAMEMODE=="a")?(3):(6)));
  }
}

# 図形データ
function MakePatternData(tbl,  k){
  k=1;
  tbl[k,1]="0100010,1000001,1000001,1000001,0100010,0011100,0000000";
  tbl[k,3]="0000000,0011100,0100010,1000001,1000001,1000001,0100010";
  tbl[k,4]="0111000,1000100,0000010,0000010,0000010,1000100,0111000";
  tbl[k,2]="0001110,0010001,0100000,0100000,0100000,0010001,0001110";
  k=2;
  tbl[k,1]="0001000,0011100,0101010,1001001,0001000,0001000,0001000";
  tbl[k,3]="0001000,0001000,0001000,1001001,0101010,0011100,0001000";
  tbl[k,4]="0001000,0010000,0100000,1111111,0100000,0010000,0001000";
  tbl[k,2]="0001000,0000100,0000010,1111111,0000010,0000100,0001000";
  k=3;
  tbl[k,1]="0010000,0010000,0011110,0010000,0010000,1111111,0000000";
  tbl[k,3]="0000000,1111111,0010000,0011000,0010100,0010010,0010000";
  tbl[k,4]="0010000,1111111,0010000,0101110,0100100,1001110,0000000";
  tbl[k,2]="0010000,1111111,0010000,0101110,0101010,1001110,0000000";
}

# 図形表示
```

```
function DispPattern(kind,dir,mode,  p,l,wk){
  print "";
  delete wk;
  split(PatternTbl[kind,dir], wk, ",");
  print "+-------------+";
  for(l=1; l<=7; ++l){
    printf("|");
    for(p=1; p<=7; ++p) printf("%s", Block[substr(wk[l],p,1)+2*mode]);
    print "|";
  }
  printf("+-------------+");
}
function DispTask(n,  sw){
  if(GAMEMODE=="a" && n>DispTaskLimit) return;
  if(n>DispTaskLimit) n=rnd(DispTaskLimit)+1;
  sw=Translate(KindTbl[0], KindTbl[n])-1;
  DispPattern((sw%3)+1,TaskTbl[n],int(sw/3));
}
function Translate(mode,val,  st,kind,r1,r2){
  st=int(mode/10);  kind=mode%10;
  if(st==1) return kind;
  if(st==2) return kind + Divide(val,2)*3;
  if(st==3 || st==4){
    if(kind==1){ r1=1; r2=2; }
    if(kind==2){ r1=2; r2=3; }
    if(kind==3){ r1=3; r2=1; }
    return ((Divide(val,2)==0)?(r1):(r2)) + ((st==4 && rnd(10)<5)?(3):(0));
  }
  if(st==5) return 1 + Divide(val,3);
  if(st==6) return val;
  return 0;
```

```
}
function Divide(val,cnt,  num){
  if(cnt==2) return int((val-1)/(6/2));
  if(cnt==3) return ((val-1)%3);
  return 0;
}

# 回答確認
function Confirm(times,level,ans,  tmp,n,ansLen,ansList){
  if(GAMEMODE=="b" || times<=level)
    printf("  #%d/%d", times, ((GAMEMODE=="a")?(level):(level+10)));
  if(times<=level){
    PushENTER();
    CLS();
    return;
  }

  print "¥n";
  if(GAMEMODE=="a"){
    printf("   ");
    for(n=1; n<=level; ++n) printf("%s", ((n%5)?("."):((n%10) "")));
    printf("[%d]¥n", level);
    ansLen=level;
  }
  if(GAMEMODE=="b"){
    printf("Which #%d ?¥n", times-level);
    ansLen=1;
  }
  InputAnswer(ansList,ansLen);
  if(GAMEMODE=="b" && times<level+10+level){
    CLS();
```

```
    printf("(#%d)¥n=> %s¥n", times-level, ansList[1]);
  }
  printf("   %s¥n¥n", Convert(ansList,ansLen));

  AnswerCheck(times,level,ansList);
  if(GAMEMODE=="a") for(n=1; n<=ansLen; ++n) ans[n]=ansList[n];
  if(GAMEMODE=="b") ans[times-level]=ansList[ansLen];
}
function InputAnswer(ans,len,  t,p){
  delete ans;
  printf("?> ");  t="";  getline t;
  for(p=1; p<=length(t); ++p) ans[p] = Key2Dir[tolower(substr(t,p,1))] +0;
  for(; p<=len; ++p) ans[p]=0;
}
function PushENTER(  tmp){ printf("  <PUSH ENTER>");  getline tmp; }

# 回答比較
function AnswerCheck(n,lev,ansList,  ans,result){
  if(n<=lev) return;

  if(GAMEMODE=="a"){
    ans=Convert(TaskTbl,lev);
    result=(Diff(TaskTbl, ansList, lev)==lev);
  }
  if(GAMEMODE=="b"){
    ans=DirStr[TaskTbl[n-lev]+0];
    result=(TaskTbl[n-lev]==ansList[1]);
  }
  printf(">> %s    %s¥n", ans, (result)?("* OK *"):("- NG -"));
  print "";
}
```

```
function Diff(tbl1,tbl2,len,  cnt){
  for(cnt=0; len>0; --len) if(tbl1[len]==tbl2[len]) ++cnt;
  return cnt;
}
function Convert(src,size,  p,dst){
  dst="";
  for(p=1; p<=size; ++p) dst = dst DirStr[src[p]+0];
  return dst;
}

# メインループ
function RoundLoop(taskLimit,offset,  st,p,r,limit,n, sc){
  limit=(GAMEMODE=="a")?(3):(6);
  for(r=1; r<=limit; ++r){
    CLS();

    delete KindTbl;
    SetSequence(KindTbl,taskLimit);
    if(GAMEMODE=="a"){ st=((Stage>6)?(6):(Stage));  p=r; }
    if(GAMEMODE=="b"){ st=r;  p=1; }
    KindTbl[0]=GetRoundPattern(st, p);
    delete TaskTbl;
    SetSequence(TaskTbl,taskLimit);
    FixSequence(TaskTbl,taskLimit);

    delete AnsTbl;
    for(n=1; n<=taskLimit+offset; ++n){
      DispTask(n);
      Confirm(n,Level,AnsTbl);
    }
    Score[Stage,r]=Diff(TaskTbl,AnsTbl,taskLimit);
```

```
    if(Score[Stage,r]!=taskLimit) break;

    print "";
    if(r<limit) PushENTER();
  }
  if(r>limit) r=limit;
  Score[Stage,0]=r;

  sc=DispResult(Stage,taskLimit,limit);
  limit*=taskLimit;
  if(sc==limit){ print ""; PushENTER(); }
  return limit-sc;
}
function GetRoundPattern(st,p){
  if(p>=3) ++st;
  if(st>6) st=6;
  if(1<=st && st<=4) return st*10 +rnd(3)+1;
  return st*10;
}
function SetSequence(list,len,  n){
  delete list;
  n=1;
  if(len>=8 && rnd(6)>4){
    if(rnd(6)<3) split("1,1,3,3,4,2,4,2", list, ",");  # 573 Command
    else         split("1,3,1,1,3,1,3,3", list, ",");  # Puppy love
    n+=8;
  }
  for(; n<=len; ++n) list[n]=rnd(6)+1;
}
function FixSequence(list,len,  n){
  if(list[1]>4) list[1]=rnd(4)+1;
```

```
  for(n=2; n<=len; ++n)
    if(list[n]>4) list[n]=((rnd(6)<1)?(list[n-1]):(rnd(4)+1));
}
function GetLevel(st,mode,  lv){
  lv=0;
  if(mode=="a"){
    if(st>6)          lv = 16;
    if(6>=st && st>3) lv = (st-2)*4;
    if(3>=st)         lv = st*2;
  }
  if(mode=="b"){
    if(st>9)  lv = 10;
    if(9>=st) lv = st+1;
  }
  return lv;
}

# 成績表示
function DispResult(st,taskLimit,limit,  n,r,sc){
  sc=0;
  r=Score[st,0];
  for(n=1; n<=r; ++n) sc+=Score[st,n];
  printf("¥n[%02d] %3s/%-2d <N=%-3s", st, "#" r, limit, Level ">");
  limit*=taskLimit;
  printf("   RESULT:%3d%%=%2d/%2d¥n", int(100*sc/limit), sc, limit);
  return sc;
}
```

---

(投稿: 2010年12月22日)

# 編集後記

jscripter

## デジタルと国境

リーマンショックから2年が過ぎ、モノを生み出す企業はグローバル化に否応なく突き進んでいく。ソフトウェア企業はどうしているんだろう。デジタルに元々国境はないか。あるのは言語的文化的障壁だけ。

## Twitter というインターフェース

アップルはハードウェアとソフトウェアとメディアのハイブリッド化を進めている。iTunes Store はメディアである。ここではコンピュータで取り扱える情報が販売あるいは配信される。音楽・オーディオブック・テレビ番組・映画・ゲーム・本・ソフトウェア・Podcast・iTunes U など。Ping は Twitter と連動するようだ。PC/Mac、iPod touch、Apple TV、テレビなどのハードウェアは iTunes、iOS や AirPlay などのソフトウェアを介してネットワークでつながる。

後を追うのは、ソニーと Google。torne/PS3 では Twitter とテレビ番組(地デジ)が連動するようになった。toren/PS3 は PSP 上のリモートでも動作する。Google TV も Android と連動するらしい。

おそらく、次はデジカメや車などがコンピュータを介してネットワークにつながるのが当然のような世界が現出するだろう。スクリプティングがどう絡めるのかが興味深いところではある。今は、Twitter がインターフェースとなる可能性について考えている。

## フリーソフトウェア

コンピューティング・ネットワークビジネスの世界の大きな発展と活況はともかく、最近のソフトウェアの世界では個人の生み出すフリーソフトウェアは Firefox のアドオンに見るぐらいだったが、フリーソフトウェアの活躍する領域はモバイルに広がってきた。App Store でフリーソフトも配信されている。もちろん、Linux やスクリプティング言語などのように、フリーソフトウェアであってもソフトウェアの基盤となるような大規模なものは、開発が個人からコミュニティに移行しているものが多い。

## Zed 0.52β

そのような世界の中でもPC上では、テキストエディタは依然として個人で開発されているものが多い。その中にZedが登場している。僕はTSNETスクリプト通信1.3に「Zed入門」を、2.2に「Zed入門 II - Twitter on Zed」、2.4に「Zed入門 III TAG実行を使ってみる」を書いてきた。スクリプトを動作させるプラットフォームとして大変有用なテキストエディタである。エディタ機能の拡張用にLuaが組み込まれていることも特筆すべきである。

最近、Zed 0.52β が出た。Zedはサブウインドウにコマンドラインを持つのだが、コマンド履歴機能が搭載された。これでZedのコマンドラインは相当使える。編集モードではコマンドライン出力を自在に編集に使えるので、Windowsのコマンドプロンプトと比べ、大変便利になった。スクリプトのデバッグなど、スクリプティングとの相性は抜群だろう。

今後、Zedのサブウインドウが UTF-8 を取り扱えるようになれば大変素晴らしいだろう。期待している。今のままでも十二分に実用的だが・・・

## 次号に向けて

デスクトップとWebの融合というコンセプトをツイッターをインターフェースにしてさらに発展させていこうと思っている。プラットフォームはデスクトップCGIとZedということになる。第12号は新年2月か3月ということでよろしくお願いします。

(投稿: 2010年12月23日)

# ＴＳＮＥＴスクリプト通信

ISSN 1884-2798　出版地: 広島市

---

2010 年 12 月 24 日　　3.3.001 版

## 投稿規程

TSNETWiki ： 「投稿規程」のページを参照のこと

## 編集委員会(投稿順)

ムムリク　qublilabo[at]gmail[dot]com
海鳥　kaityo256[at]gmail[dot]com
jscripter　jscripter9[at]gmail[dot]com
Ｙさ　saw[at]mf-nokuchi2pho[dot]ne[dot]jp

## 著作権



## 使用許諾・配布条件

## 免責事項

「TSNETスクリプト通信」の内容および同時に配布されるスクリプトなどの使用は、すべて使用者の自己責任によるものとし、使用によって生ずる一切の結果等について、編集委員会および著作者は責任を負いません。

## 編集ソフトウェア

OpenOffice.org 3.2.1 Writer

## 発行所

一次配布所: TSNETスクリプト通信刊行リスト

http://text.world.coocan.jp/TSNET/?TSNET%E3%82%B9%E3%82%AF%E3%83%AA%E3%83%97%E3%83%88%E9%80%9A%E4%BF%A1%E5%88%8A%E8%A1%8C%E3%83%AA%E3%82%B9%E3%83%88

TSNETスクリプト通信 第3巻第3号(通算第11号)
発行: TSC編集委員会 発行日: 2010年12月24日
ISSN: 1884-2798 出版地: 広島市 創刊: 2008年5月7日