コラム4　パーソナルコンピュータを用いた心理学実験　補足資料

1 「連想プライミング」プログラムの解説[1]

1.1 プログラムのコンパイルと実行

HSP のホームページ（下記，「2 参考 Web ページ」）から最新版（2015 年 7 月現在では 3.4）のインストーラをダウンロードして実行する。その後，プログラムのソースファイル（"連想プライミング.hsp"）をダブルクリックすると，HSP スクリプトエディタが起動してファイルが読み込まれる。メニューバーから，「HSP(P)」→「コンパイル+実行(C)」とすると，プログラムが翻訳・実行される。また，「HSP(P)」→「実行ファイル自動作成(A)」とすると，実行ファイル（"hsptmp.exe"）が，カレントディレクトリに作成される。このファイルをコピー（および必要に応じてリネーム）すると，他のフォルダや，HSP をインストールしていない他の PC で実験を行うことができる。ただし，実験刺激のファイル（"リスト練習用.csv" と "リスト 1.csv"，"リスト 2.csv"）を実行ファイルと同じフォルダに置く必要がある。

以下では，ソースファイルの一部を引用しながら，心理学実験プログラミングのポイントを解説する。なお，初心者のための可読性を優先し，効率性や高速性を多分に犠牲にしたコードである点に留意されたい。左端の数字は，行番号を表す（プログラムには含まれない）。HSP スクリプトエディタ以外のエディタで行番号を確認する場合は，「エディタ的（改行だけを数える）」設定で表示する。

HSP の文法や命令の詳細については，HSP スクリプトエディタのメニューの「ヘルプ(H)」や 本文書「2 参考 Web ページ」で紹介する HSP 入門サイト，書籍本文の参考文献などを参照のこと。

1.2 ミリ秒タイマーの準備

```
3:      ; ミリ秒タイマー準備
4:      #uselib "winmm.dll"
5:      #func timeBeginPeriod "timeBeginPeriod" sptr
6:      #func timeEndPeriod "timeEndPeriod" sptr
7:      #cfunc timeGetTime "timeGetTime"
8:
9:      timeBeginPeriod 1     ; タイマーの精度を 1ms に設定
…
833:    timeEndPeriod 1       ; タイマーの精度を復帰
```

「;」または「//」から行末まではコメントで，プログラムの実行には影響しない(3 行)。

Win32 API の機能を呼び出してミリ秒タイマー関数（timeGetTime）を使えるようにするための準備(4～7 行)を行う。タイマーを実際に使用する前に，"timeBeginPeriod 1" として，タイマーの精度を 1ms に設定する(9 行)。プログラム終了前に，"timeEndPeriod 1" として，タイマーの精度を復帰する

[1] プログラムのソースファイルと実験刺激のファイルは，「13 章 連想プライミング」のサポートサイトにある。

(843 行)。なお，より高い時間精度が必要なら，「2 参考 Web ページ」で紹介している外部モジュール（TimeManager, alib_wait など）の使用を検討すべきである。

1.3 画面の初期化

```
12:     mouse -1                                ; マウスカーソル非呈示
13:     画面幅 = ginfo_dispx
14:     画面高 = ginfo_dispy
15:     bgscr 0, 画面幅, 画面高, 0, 0, 0        ; 枠なしでディスプレイ全面初期化
16:     cls 4                                   ; 黒で画面クリア
```

マウスカーソルを非表示にし(12 行)，ディスプレイ全面を初期化する(15 行)。

ginfo_dispx, ginfo_dispy はシステム変数で，プログラム起動時に，ディスプレイの X 方向の画像サイズと Y 方向の画像サイズが，それぞれ自動的に設定される。

1.4 刺激ファイルの読み込み[2]

```
49:     刺激ファイル = "リスト 1.csv"
…
131:    sdim 刺激ノート
132:    notesel 刺激ノート
133:    noteload 刺激ファイル
134:    sdim 試行 ID
135:    sdim プライム
136:    sdim ターゲット
137:    sdim タイプ
138:    試行数 = notemax
139:    repeat 試行数
140:            noteget 行, cnt
141:            sdim 項目
142:            split 行, ",", 項目
143:            試行 ID(cnt) = 項目(0)
144:            プライム(cnt) = 項目(1)
145:            ターゲット(cnt) = 項目(2)
146:            タイプ(cnt) = 項目(3)           ; R=関連あり/U=関連なし/N=無意味
147:     loop
```

[2] 刺激ファイルには，"リスト 1.csv" と "リスト 2.csv" の 2 種類がある。ここでは，"リスト 1.csv" を選択して用いる場合について説明している。

まず sdim 命令で，後で必要になる文字列型配列を宣言する(131,134～137 行)。
刺激単語は，以下のような CSV ファイル（1 行＝1 レコード, カンマで値を区切ったテキストファイル）から，HSP 独自のメモリノートパッド（改行で値を区切ったテキスト）形式を経由して読み込む。

```
関連あり A01,うろこ,さかな,R
関連あり A02,おじぎ,れいぎ,R
…省略…
無意味 60,いびき,すりも,N
```

"リスト 1.csv"

１行目を例にすると，"関連あり A01" が試行 ID，"うろこ" がプライム，"さかな" がターゲット，"R" が試行のタイプにあたる。

notesel 命令で以後の処理対象となるノートパッドを指定し(132 行), noteload 命令で指定したファイルの内容をノートパッドに読み込む(133 行)。その際，読み込まれた行数がシステム変数 notemax に設定される(138 行)。

repeat ～ loop は，repeat と loop の間の処理を，repeat の引数で指定された回数だけ繰り返す(139～147 行)。その際，現在のループ回数がシステム変数 cnt に設定される（1 ではなく 0 から始まることに注意)。ループ内では，ノートパッドから 1 行を読み込んでは(140 行)，カンマ区切りごとに分割し(142 行)，分割された要素を対応する配列に読み込む(143～146 行)。

1.5 試行順序のランダマイズ

```
172:    // 本番試行順序のランダマイズ
173:    dim 試行順, 120
174:    repeat 試行数
175:            試行順(cnt) = cnt
176:    loop
177:    randomize
178:    repeat 試行数
179:            待避 = 試行順(試行数 - 1 - cnt)
180:            置換 = rnd(試行数 - cnt)
181:            試行順(試行数 - 1 - cnt) = 試行順(置換)
182:            試行順(置換) = 待避
183:    loop
```

Fisher-Yates 法のアルゴリズムを用いて，ランダマイズされた呈示順序が入った配列（"試行順"）を用意する。たとえば，"試行順(3)" の値が 0 であれば，4 回目のループで 0 番目の刺激（関連あり A01）

を呈示する。

まず，試行数分の要素を入れる配列を準備する(173 行)。次に，"試行順(0)" に 0，"試行順(1)" に 1，…を順に代入する(174～176 行)。randomize 命令で乱数の発生パターンを初期化する(177 行)。

ランダマイズは，指定した配列の値を，ランダムに選んだ配列の値と交換することで行う(178～183 行)。1 回目のループでは，まず配列の最後の要素("試行順(119)")を指定して，値を待避する(179 行)。次に，指定した要素("試行順(119)")から最初の要素("試行順(0)")までのどれか 1 つをランダムに選んで(180 行)，その値を待避した値と交換する(181～182 行)。rnd 命令は，0 から 引数-1 までの整数（0～119）の乱数値を返す。以上の手順を指定の要素を 1 つずつ前にずらしながら配列の要素の数だけ繰り返す。

1.6 刺激の呈示

```
20:     プライム呈示時間 = 20          ; 200ms
…
30:     刺激フォント = "ＭＳ ゴシック"
31:     刺激サイズ = 48
…
511:    repeat 試行数   ; 本番試行メインループ
512:
513:            カウンタ = cnt  ; 試行カウンタ
…
526:            // プライム準備
527:            buffer 2              ; バッファ 2 に書き込み
528:            cls 4
529:            color 255, 255, 255
530:            font 刺激フォント, 刺激サイズ
531:            mes プライム(試行順(カウンタ))
532:            プライム幅 = ginfo(14)
533:            プライム高 = ginfo(15)
…
580:            // プライム呈示
581:            起点 x = (画面幅 - プライム幅) / 2
582:            起点 y = (画面高 - プライム高) / 2
583:            redraw 0
584:            pos 起点 x, 起点 y
585:            gcopy 2, 0, 0, プライム幅, プライム高  ; バッファ 2 からコピー
586:            redraw 1
587:
```

```
588:            wait プライム呈示時間
…
700:    loop    ; 本番試行メインループ
```

プライム単語の呈示を例に解説する。注視点やプライム，ターゲット等の呈示時間や，フォントや文字サイズの指定などは，プログラムの冒頭部分でまとめて設定しておくのが良い。時間の指定については，10ms 単位の命令（wait）と 1ms 単位の命令（await）があるので注意すること。

文字や画像などを呈示する際には，現在表示している画面に直接書き込むのではなく，バッファ（裏画面）に書き込んだ内容を，表示画面にコピーする。プログラムでは，バッファ 1 を注視点，バッファ 2 をプライム，バッファ 3 をターゲット…，に使用している。buffer 命令で，書き込みの対象にするバッファを指定し(527 行)，mes 命令でプライム文字列を描画する（531 行，ディスプレイにはまだ表示されない)。この際，描画した文字列の X 方向と Y 方向のサイズを保存しておく(532～533 行)。

プライムを呈示する際には，保存した文字列のサイズを利用して，画面の中央に表示されるように描画の起点を決める(581～582，584 行)。バッファから表示画面へのコピーは，gcopy 命令で行う(585 行)。その際に，画面のちらつきを防ぐために，"redraw 0" としていったん画面表示の更新を停止し(583 行)，コピー実行後に "redraw 1" として画面表示を再開する(586 行)。

なお，画面の表示に関しては，es_buffer や es_copy といった DirectX を利用した命令を用いた方が高速である。

1.7 反応時間の取得

```
607:    // 反応時間計測スタート
608:    スタート = timeGetTime()
609:
610:    // キー押しチェック
611:    repeat
…
641:            getkey 左キー, 37               ; 「←」キー(「単語」判断)
642:            if 左キー {
643:                    ストップ = timeGetTime()        // 計測ストップ
644:                    if タイプ(試行順(カウンタ)) = "N" : 誤反応(試行順(カウンタ)) = 1
645:                    break
646:            }
647:            getkey 右キー, 39               ; 「→」キー(「非語」判断)
648:            if 右キー {
649:                    ストップ = timeGetTime()        // 計測ストップ
650:                    if タイプ(試行順(カウンタ)) ! "N" : 誤反応(試行順(カウンタ)) = 1
651:                    break
```

```
652:            }
653:            getkey エスケープ, 27  ; 「ESC」キー
654:            if エスケープ : end
655:
656:            await
657:    loop
…
665:            // 反応時間計算
666:            時間差 = ストップ - スタート
667:
668:            // 反応保存
669:            反応時間(試行順(カウンタ)) = 時間差              ; ミリ秒単位
670:            反応キー(試行順(カウンタ)) = 左キー              ; 単語なら 1, 非語なら 0
```

反応時間は，計測開始時(608 行)とキー反応時(643, 649 行)に timeGetTime 命令でタイマー起動からの経過時間を記録し，両者の差を計算して求める(666 行)。試行順の配列を用い，刺激ファイルの並び順で反応保存用の配列に保存する(669～670 行)。

repeat ～ loop は無限ループになっており，左矢印キー（「←」），右矢印キー（「→」）のいずれかが押されると break でループから抜ける（ESC キーが押されるとプログラムを終了する)。キーが押されたかどうかの判断は，getkey 命令で行う。"getkey 変数, キーコード" で，キーコードに対応するキーが押されると，変数に 1 が代入される。続けて誤反応かどうかの判定と保存を行う(644, 650 行)。

1.8 反応データをファイルに保存

```
50:     反応ファイル = "反応 1_"
…
704: // 反応データ列を作成
705:
706:    反応 = ""
707:
708:    // CSV ヘッダー
709:    反応 += "試行 ID" + "," + "タイプ" + "," + "反応時間" + "," + "反応キー"
             + "," + "誤反応" + "¥n"
710:
711:    // 反応データ
712:    repeat 試行数
713:            反応 += 試行 ID(cnt) + "," + タイプ(cnt) + "," + 反応時間(cnt)
                    + "," + 反応キー(cnt) + "," + 誤反応(cnt) + "¥n"
```

```
714:    loop
715:
716:    // ファイル名後半(終了時点の日時情報)
717:
718:    month  = str(gettime(1))
719:    day    = str(gettime(3))
720:    hour   = str(gettime(4))
721:    minute = str(gettime(5))
722:    second = str(gettime(6))
…
728:    ファイル名後半 =  month + day + hour + minute + second + ".csv"
729:
730:    // 反応をファイルに書き出し
731:    notesel 反応
732:    反応ファイル += ファイル名後半
733:    notesave 反応ファイル
```

反応データファイルを出力するには，項目をカンマ（「,」）で区切り，レコード（1 回分の試行データ）を改行（「¥n」）で区切った CSV 形式の長い文字列（"反応"）を作成し，メモリノートパッド形式を経由してテキストファイルに出力する。まず，反応データを入れる文字列を初期化し(706 行)，1 行目にヘッダ（709 行），2 行目以降に反応データを追加する(713 行)。文字列データに加算演算子「+」を適用すると，文字列が結合される。反応取得の際に，刺激ファイルの並び順に直してデータを格納しているため，ここで再びデータを並べかえる必要はない。

出力ファイル名は，重複を避けるために実験実施時の日時情報を用いている(718～728，732 行)。たとえば，7 月 31 日 13 時 12 分 50 秒であれば，"反応 1_07 月 31 日 13 時 12 分 50 秒.csv" となる[3]。notesel 命令で対象となるノートパッドを指定し，notesave 命令でテキストファイルに保存する。

2 参考 Web ページ

※敬称略，2015 年 7 月 31 日現在

・HSP 公式ページ

現在のバージョンは 3.4

http://hsp.tv/

・HSP 入門

[3] ファイル名の "反応 1_" の "1" は，刺激ファイルとして "リスト 1.csv" を選択したことに対応している。また，ファイル名は重複しなければよいので，実験終了後，適当に変更してよい。

コラム４　パーソナルコンピュータを用いた心理学実験　補足資料

http://hsp.tv/make/enroll.html

・Time Manager for HSP3
HSP3 用高精度時間管理モジュール
http://sakmis.6.ql.bz/module.html#tmanage3

・alib_wait for HSP3.1
HSP3.1 用高精度ウェイトモジュール
http://ayaoritomoe.oiran.org/alib_wait/

・演算スパンテスター
井関龍太（京都大学情報学研究科）
HSP で書いた演算スパン（operation span）テストのプログラム
http://riseki.php.xdomain.jp/index.php?演算スパンテスター

・python で心理学実験
十河宏行（愛媛大学）
PsychoPy Coder を用いた心理学実験
http://www.s12600.net/psy/python/index.html

・PsychoPy 講座
小川洋和（関西学院大学）
PsychoPy Builder を用いた心理学実験
http://ogwlab.org/?page_id=460

・心ポ（心理学ポータル）ツール
心理学の研究・教育に役立つツールを紹介するページ
http://www27.atwiki.jp/simpo/pages/18.html

・心理学ワールド 67 号［小特集］プログラミング THE 心理学実験
PsychoPy, Psychlops, Psychtoolbox, Processing
http://www.psych.or.jp/publication/world067.html

・PEPE プログラミング THE 心理実験
心理学ワールドの 67 号の心理学実験プログラミング小特集関連ページ
http://www.odalab.org/pepe/

・Psychtoolbox をがんばる：心理学、実験、プログラミング

コラム４　パーソナルコンピュータを用いた心理学実験　補足資料

https://sites.google.com/site/ptbganba/

・Psychlops wiki
http://psychlops.osdn.jp/ja/

・jsPsych
Web ブラウザ上で心理学実験を行うための JavaScript ライブラリ
http://www.jspsych.org/