（一部、HSP2.61 の表記法のスクリプトが残っていることがあるかも）

HSP

英語教育研究のためのプログラミング：HSP の基礎テキスト

HSP で教材開発

HSP で言語心理学的実験

HSP の導入

■ダウンロードとインストール

https://hsp.tv/idman/download.html

■ HSP スクリプトエディタの設定

・「ツール」＞「オプション」
　・「エディタ」＞「表示」
　　・「非文字の可視化」で、「TAB」「半角スペース」「全角スペース」「改行」にチェックを入れる

コメントアウト

```
; で、その行のそれ以降コメントアウト

/*
この上下で挟まれた部分は、複数行にわたってコメントアウト
*/
```

画面「消去」の３つの命令

1. boxf
   ・ 画面（のある範囲）を「色」で塗りつぶすことで「消去」する
2. cls
   ・ 画面上のすべてを初期化してしまう
3. screen
   ・ ウィンドウの初期化

参考：https://wiki.hsp.moe/

画像の一部分のコピー（画像を「部品」として使う）

・gmode によってコピーするモードを選ぶ
・gcopy によって、どのウィンドウ ID の画像のどの部分をコピーするか指定

```
gmode 0
gcopy ウィンドウ ID，コピー元座標 x，y，x 軸で何ドット分か、y 軸で何ドット分か
```

PictureFeedback 参照

文字列と数値の連結

・数値の文字列変換： str()

```
str(suuji)+" もじもじ " => 文字列
```

・文字列 + 数値型変数

```
" もじもじ "+suuji => 文字列
```

```
「文字列型」に「数値型」を足し算した場合には、「数値」を文字列にして連結される。
「数値型」に「文字列型」を足し算した場合には、「文字列」を数値にして加算される。
つまり、計算式で最初の項になっている型に自動的に合わせてくれるということです。
```

http://www.onionsoft.net/hsp/v34/doclib/hsp3str.htm

文字の改行

■文字列中に \n を入れることで、そこで改行される。

■適当に改行したうえで、前後の引用符をさらに { } で囲む。

音声を録音する

■録音の設定

・set を分けて書くと、うまく動きません。音質が悪くなります。

・以下のようにすると、音声が悪くなります。

```
mci "set sam channels 2"
mci "set sam samplespersec 44100"
mci "set sam bitspersample 16"
mci "open new alias sam type waveaudio"
```

■録音の開始・停止・終了

```
mci "record sam"        ; 録音の開始
mci "stop sam"          ; 録音停止
mci "save sam \"sample.wav\""      ; 音声ファイル保存
mci "close sam"          ; 録音終了（デバイスを閉じる）
```

■ (2020-06-14)

上記のスクリプトの 2 行目「samplepersec」ではなく「samplepersec」です。

間違えていました。すみません。

■一行に

・mci 命令は Windows 7 以降、一行にまとめて書かないといけなくなったようです。

http://hsp.tv/play/pforum.php?mode=pastwch&num=39979

■サンプルプログラム

sound recording

動画の再生

■マルチメディア制御

http://lhsp.s206.xrea.com/manual/i_mmedia.html

■ mci 命令

https://blog.goo.ne.jp/predater/e/7b8985aa3cea94896fd61bebd5c32098
http://lhsp.s206.xrea.com/command/mci.html
https://docs.microsoft.com/ja-jp/windows/win32/multimedia/mci

実行ファイル作成時にファイル名を特定しておく

```
#packopt name "postoffice01"
```

実行ファイル作成時に独自のアイコンを設定する

```
#packopt icon " ファイル名 .ico"
```

■ ico 形式の画像ファイルを作成して、同じフォルダーに入れておく

実行ファイル作成時に複数のファイルをまとめてコンパイルする

```
#pack " ファイル名 "
```

・pack できるのは
　・画像ファイル *.jpg
　・テキストファイル *.txt
　・音声ファイル *.wav
　　・ただし、大きさは 2MB 以下
　・ファイル名は 15 文字以下にすること

文字を順番に表示する

```
text ミリ秒
mes " 文字列 "
```

text で設定したミリ秒ずつ wait がかかって mes で指定した文字列が順に表示されていく。

■ HSP3 では、#include "hsp3util.as" して、mes ではなく emes を使うことになったの

で注意。

データを保存する

notesel 変数名 ; データを保存する特別な変数を設定
noteadd " データ " ; データを追加
notesave " ファイル名 " ; データを「ファイル名」で保存する

オブジェクト

ボタンの大きさを変える objsize 横 , 縦

・デフォルトは 64 x 24

```
objsize 200, 20
button " ちょっと長い名前 ", *nextstep
```

配列 （配列変数）

複数の変数を順に並べて番号を付けたもの。
番号で変数を指定できるので、順番に処理するのに便利。

変数と同様に、配列に名前をつける。(kamoku)
名前のうしろに括弧で番号をつける。
番号は 0 から始まる。

```
kamoku(0)
```

値は変数と同様に = で代入する。(「配列変数 kamoku の要素 0 に 39 を代入する」という)

```
kamoku(0) = 39
kamoku(1) = 29
kamoku(2) = 32
```

■配列の数を調べる length(配列名)

```
kamokusuu = length(kamoku)
```

■ HSP2.61 でのこと

```
配列の後ろに、番号を付ける。(kamoku.0)
１５個以上使うときは、最初に dim で使う数を設定する。
dim kamoku,20

文字を扱うときは、sdim で設定する。
sdim 配列名 , 文字数 , 要素数

ひとつの科目名の文字数は２０文字以下で、科目は１０科目あるとすると、
sdim kamoku,20,10
```

テキストを読み込む

```
notesel 変数名
noteload " ファイル名 "
mes 変数名
```

変数名は何でも良い。bun とか、、、
一行が長い場合、自動的に改行はしてくれないので注意。

■読み込むファイル形式と文字コードに注意

・Excel の csv ファイルは文字コードに注意
　・UTF-8(BOM あり) だと、ファイルの先頭に BOM という特殊な文字が入り込んでしまう。
　　・https://qiita.com/yamashiroakihito/items/6922661fb6d5753b67d8
　　・https://qiita.com/mitsutoyo32104/items/76335bb6e783d55f311c
・エディターで、テキストファイルとして編集して、文字コードを確認して、拡張子を .txt にして読み込んだ方が安心。
　・SJIS にしておくのが安全。

画像を読み込む

```
picload "ファイル名"
```

ウインドウサイズは、読みこんだ画像の大きさになる。

事前にウインドウを表示して、その中に画像を位置づけるには、

```
pos 100, 200
picload "ファイル名", 1
```

dialog を使ってテキストファイルを読み込む

```
dialog "拡張子", 16, "補足説明"
```

・dialog のオプション 16 で、ファイル選択のウィンドウが開く
・読み込んだファイル名は refstr というシステム関数に保存される。
　・ファイル名は絶対パスでドライブ名・ディレクトリー名付き

dialog を使って音声ファイルを読み込み再生する

dialog を使って動画ファイルを読み込み再生する

getkey　と　stick の違い

getkey は連続してキーの値を受け付ける。
stick はキーを押したときに一度だけ値を受け取る。

```
repeat
;  getkey a, 32    ; 特定のキーを先に想定
;  if a&1 : break  ; a には、そのキーが押されたという意味の「1」が入っている。

  stick a, 0      ; 何が押されるかはわからない
  if a&16 : break ; a には押されたキーのコードが入っている。

loop

mes "Hit"

stop
```

キーボードやマウスボタンのコード （stick）

「ヘルプ」の「HSP 命令リファレンスを開く」で stick の項目を見る。

```
  1 : カーソルキー左 ( ← )
  2 : カーソルキー上 ( ↑ )
  4 : カーソルキー右 ( → )
  8 : カーソルキー下 ( ↓ )
 16 : スペースキー
 32 : Enter キー
 64 : Ctrl キー
128 : ESC キー
256 : マウスの左ボタン
512 : マウスの右ボタン
1024 : TAB キー
```

キーボードやマウスボタンのコード （getkey）

「ヘルプ」の「HSP 命令リファレンスを開く」で getkey の項目を見る。

```
    1    : マウスの左ボタン
    2    : マウスの右ボタン
    3    : キャンセル（[CTRL]+[BREAK]）
    4    : ３ボタンマウスのまん中のボタン
    8    : [BACKSPACE]（PC98 の [BS]）
    9    : [TAB]
   13    : [ENTER]
   16    : [SHIFT]
   17    : [CTRL]
   18    : [ALT]（PC98 の [GRPH]）
   20    : [CAPSLOCK]
   27    : [ESC]
   32    : スペースキー
   33    : [PAGEUP]（PC98 の [ROLLDOWN]）
   34    : [PAGEDOWN]（PC98 の [ROLLUP]）
   35    : [END]（PC98 の [HELP]）
   36    : [HOME]（PC98 の [HOMECLR]）
   37    : カーソルキー [ ← ]
   38    : カーソルキー [ ↑ ]
   39    : カーソルキー [ → ]
   40    : カーソルキー [ ↓ ]
48 ～ 57   : [0] ～ [9]（メインキーボード）
65 ～ 90   : [A] ～ [Z]
```

動きを一時停止する

```
wait 300 ; 300 x 10 ミリ秒 = ３秒　待つ
```

画面の色を変える

＝画面と同じ大きさの四角を、色で塗りつぶす

```
color 0,255,255 : boxf 0,0,640,480
```

color コマンドで色を指定し、直後に、boxf コマンドで範囲を指定する。

boxf: 四角い範囲を指定して色で塗りつぶす。
四角い範囲＝ウィンドウの大きさ　にすると画面全体の色が変わる。

boxf は、範囲を指定しなければ、ウインドウ全体を塗りつぶす。だから、結論としては、画面の色を変えるには、

```
color 0,255,0 : boxf
```

とするだけでよい。

色を設定する

■ RGB 値で指定する

```
color 0,200,0
color Red, Green, Blue
```

・0(暗い) - 255(明るい) で値を決める。
・color だけで数字を指定しないと、黒になる

■文字列の一部だけ色を変える

・color で文字の色を指定し
・mes 命令のオプション 1 で改行しないようにする

```
mes "ここは黒",1 :color 255, 0, 0: mes "ここは赤", 1: color: mes "後ろを黒に"
```

フォントを指定する

```
font "ＭＳ ゴシック", 32, 1
```

```
font "フォント名", サイズ , スタイル
```

スタイルの設定

```
1 太字
2 イタリック
4 下線
8 打ち消し線
16 アンチエイリアス
```

スタイルを組み合わせたい場合は、数字を足す。

位置を指定する

```
pos 左上横 , 左上縦
```

ウィンドウの広さはデフォルトでは 640 x 480

実際に実験する際に注意すること

プログラムの実行中にＨＳＰのプログラム以外のものが動いたりして、測定に影響を与えるといけないので、以下の点に注意すること。
１）スクリーンセーバーを切っておく
２）電源の設定で、節電にしないこと。（処理能力最優先にする）
３）ネットワークの接続をすべて無効にしておく。
４）ウイルスチェックのプログラムを止めておく。

５）WindowsUpdate などの自動更新を止めておく。
６）画面の色を必要最低限にする。
７）デスクトップの背景に何もないようにする。

主なコマンド

■ str

変数を文字列型に変更。
オプションで、ゼロを含めた桁数にできる。

```
str 数値を含む変数 , オプション
```

```
suuji= 7
str suuji, 2
```

これにより suuji には「07」という文字列が入る。

■ strlen

文字列の長さを調べる。

```
strlen 結果の変数名 , 調べたい文字列
```

全角は 2 文字分。

■ strmid

・部分的に文字列を取り出す。

何文字目から何文字取り出すか、と指定する。

```
strmid 結果を保存する変数 , 取り出し元の変数 , 何文字目から , 何文字分を
```

```
text='When I was a child, I lived in Tokyo.'
strmid kekka, text, 7, 3
```

kekka には、「was」が入る。
文字の位置は 0 から始まる。７と指定すると 8 文字目から。

・特定の文字列以降を取り出すには

```
text='When I was a child, I lived in Tokyo.'
instr koko, text, "," : getstr kekka, text, koko+2
```

カンマ以降をとりだし、kekka には「I lived in Tokyo.」が入る。
前半で instr でカンマの位置を調べて、後半で getstr でカンマの位置に直後のスペース一文字分を足して「2」を足した位置から行末までを kekka に保存する。

■ instr

検索文字列が何文字目にあるかを調べる。

```
instr 結果を保存する変数 , 検索対象の変数 , " 検索文字列 ", 何文字目から検索するか
```

第 4 項目の「何文字目から検索するか」を指定しなければ行頭から。
結果は、数字で保存される。1 文字目が「0」となる。
文字列がなければ「-1」が結果となる。

```
text='When I was a child, I lived in Tokyo.'
instr kekka, text, "was"
```

kekka は「7」となる。

■ getstr

文字列を取り出す。

```
getstr 結果を保存する変数 , 取り出し元の変数 , 何バイト目から , どこまで
```

第 4 項目の「どこまで」を指定しなければ行末まで。
「どこまで」は区切りとする文字で指定する。例：','

```
kekka
text='When I was a child, I lived in Tokyo.'
getstr kekka, 0, text, ','
```

text 中の 0 文字目からカンマまでを取り出して、kekka に入れる。
kekka には「When I was a child」が入る。

■ gettime　時間の情報を得る

```
変数 = gettime ( パラメタ)
```

パラメタで、時間のどの情報を得るかを設定

```
0  年
1  月
2  曜日
3  日
4  時
5  分
6  秒
7  ミリ秒
```

現在、何時　何分　何秒か？

```
ji = gettime(4)
fun = gettime(5)
byo = gettime(6)
mes " 今は "+ji+" 時 "+fun+" 分 "+byo+" 秒です "
```

■ objimage　画像をボタンにする

```
;
```

```
  ;   画像をボタンにする
  ;
  buffer 1          ; 画像を事前に読み込んでおく場所の番号
  picload "flower1.jpg"     ; 画像を読み込んで、buffer に入れておく

  screen 0,640,480      ; ウインドウの初期化　0番、ウインドウの大きさの指定
                ; buffer を使って画像を読み込んだときは、明示的にウィンドウを指定する

  pos 300, 200          ; ボタンの位置
  objsize 180,180         ; ボタンにする画像のサイズ（大きな画像を読み込んでおいて、その一部を
                ;   ボタンとして使用することもできる。）
  objimage 1            ; ボタン画像の指定をする。最初の番号が buffer 番号

  ;objimage 1, 0,0, 0,1, 0,2  ; ボタン画像の指定をする。
                ; 最初の番号が buffer 番号
                ; 次の数字の組み合わせが、表示する画像の中の左上の位置
                ; その次の数字の組み合わせが、ボタンを押したときの位置
                ; 最後の数字の組み合わせが、ボタンの上にマウスが来たときの位置

  button gosub "", *say     ; ボタン命令で、上で設定した画像がボタンとして使われる。

  stop

*say
  mes "Hello!"
  return
```

■絵を押すと音が鳴る

```
; 透明ボタンと音声再生 objimage-sound.hsp
; copyleft 2022-07-14 sugiura@nagoya-u.jp

  ;   画像をボタンにする

#pack "left.mp3"
#pack "right.mp3"
#pack "flower1.jpg"

bcopy "left.mp3", "left.mp3"
bcopy "right.mp3", "right.mp3"

  ; 音声ファイル読み込み
  mmload "left.mp3", 1
  mmload "right.mp3", 2



  ;
  buffer 1          ; 画像を事前に読み込んでおく場所の番号
  picload "flower1.jpg"     ; 画像を読み込んで、buffer に入れておく
```

```
screen 0,640,480       ; ウインドウの初期化　０番、ウインドウの大きさの指定
                ; buffer を使って画像を読み込んだときは、明示的にウィンドウを指定する

pos 100, 200          ; ボタンの位置
objsize 90,180          ; ボタンにする画像のサイズ（大きな画像を読み込んでおいて、その一部を
                ;   ボタンとして使用することもできる。）
;objimage 1             ; ボタン画像の指定をする。最初の番号が buffer 番号

objimage 1, 0,0, 0,1, 0,2  ; ボタン画像の指定をする。
                ; 最初の番号が buffer 番号
                ; 次の数字の組み合わせが、表示する画像の中の左上の位置
                ; その次の数字の組み合わせが、ボタンを押したときの位置
                ; 最後の数字の組み合わせが、ボタンの上にマウスが来たときの位置

button gosub "", *left    ; ボタン命令で、上で設定した画像がボタンとして使われる。


pos 300, 200
objsize 90, 180
objimage 1, 90,0, 90,1, 90,2
button gosub "", *right


stop

*left
  mmplay 1
  return

*right
  mmplay 2
  return
```

■ rnd() ランダムな数字

二種類の画像のボタンを使うには

・事前に二種類の画像を合わせた一枚の画像を用意する。
・二種類の画像を左右に並べたとして、
・左側と右側の画像をそれぞれ左上隅の座標で指定することによって、
・二つの画像のボタンを配置したように表示できる。

チェックボックスでアンケート

・チェックするだけ

・自由記載欄とチェックボックス、結果をテキストファイルに保存。

システム変数

■ cnt

・デフォルトでは、0 から始まる。
　・あらかじめ指定しておけば、そこから始まる。

モジュール（開発中）

1. 被験者の入力 ID を保存ファイル名にする方法
2. 問題がランダムに出るようにする方法
3. 選択問題（ラジオボタン）
4. 自由記載ボックス

CuteHSP

開発環境

■ VSCode

https://qiita.com/umuy/items/8e8e6f1e4ccc0387be2c

スマホで HSP

■ HSPDish

https://sites.google.com/site/simakuroneko/home/hsp3dish_course/env_const/page_1

参考サイト

キーワード一覧

https://wiki3.jp/Basicsoft/page/425