

```

1  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2  ;;
3  ;;  NILSEN-DEV.LISP By Anthony Hornof (ajh) , based on Dave Kieras' example.
4  ;;  Created 9/7/95
5  ;;  Last modified 11/13/95
6  ;;
7  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
8
9  "
10 This file provides the task environment for the Nilsen menu selection experiment.
11 The experiment is Experiment #2 detailed in Erik Nilsen's dissertation entitled
12 'Perceptual Motor Control In Human-Computer Interaction',
13 which is identical to CSMIL Tech. Report #37.
14
15 The task environment was built on Dave's mini-model, which did the following:
16 the following is a little model for a simple choice RT task.
17 put up a ready signal (a cross) for one second and provide a tone signal as well
18 have a delay of half a second
19 then put up a digit that is black in color to the right and below the ready
20 signal then wait for a keystroke
21 check that the keystroke matches the digit, and update an average RT
22 change the color of the digit to white for half a second
23 then remove the digit
24 wait for another second, and then repeat
25
26 Anthony Hornof modified the following:
27 6/23/95 split STIMULUS-DIGIT-VALUE and STIMULUS-DIGIT-OBJECT into 1 and 2,
28     changed procedures to deal with both objects, such as DO-DISPLAY-STIMULUS.
29     Added LEFTMOST property in DO-DISPLAY-STIMULUS function.
30 7/10/95 modified DO-TRIAL-STATS so it prints as it goes.
31
32 7/23/95 converted *1-STIMULUS-DIGIT-VALUE and *1-STIMULUS-DIGIT-OBJECT
33     by removing the 'DIGIT-' and adding a 3rd, 4th, and 5th stimulus.
34     Added TARGET-STIMULUS, which corresponds to which object the subject
35     is primed for outside of the simulation.
36 11/13/95 - Note that I have changed Dave's methodology for triggering stimuli.
37     How:
38
39 2/23/96 - Add function that takes in distance between menu items and distance
40     from screen and returns visual angle.
41     - Make target item appear in the device.
42     - renamed DO-DISPLAY-STIMULUS to DISPLAY-MENU
43 5/31/96 - Have the device verify correct mouse locations.
44     - Have the device look for mouse up and down, not punches.
45
46 "
47 ;*****
48 ;Menu Selection Models Built Using
49 ;The EPIC Simulation Software
50 ;
51 ;Copyright (c) 1998
52 ;The Regents of the University of Michigan
53 ;ALL RIGHTS RESERVED
54
55 ;Permission is granted to use this software for noncommercial
56 ;education and research purposes and to modify this software for
57 ;such purposes only. Permission to copy or distribute this
58 ;software or any modification is not granted.
59 ;
60 ;This software is provided as is, without representation as to its
61 ;fitness for any purpose, and without warranty of any kind, either
62 ;express or implied, including without limitation the implied
63 ;warranties of merchantability and fitness for a particular
64 ;purpose. The Regents of the University of Michigan shall not be
65 ;liable for any damages, including special, indirect, incidental,
66 ;or consequential damages, with respect to any claim arising out of
67 ;or in connection with the use of the software, even it it has been
68 ;or is hereafter advised of the possibility of such damages.
69 ;
70 ;The authors of this software expect that the proper attribution
71 ;will be made in any public reports of work conducted with this

```

```

72 ;software. The report should distinguish between results due to the
73 ;architecture represented in this software and the task-specific encodings,
74 ;parameter estimates, and production-rules used to construct a model
75 ;using this software. It should also include the date or version of this
76 ;software that was used, and identify any substantive modifications.
77 ;
78 ;This menu selection simulation is under development and may change prior
79 ;to publication of this research.
80 ;
81 ;Anthony J. Hornof, Artificial Intelligence Laboratory
82 ;Electrical Engineering and Computer Science Department
83 ;University of Michigan, Ann Arbor, MI 48109-2110
84 ;hornof@eecs.umich.edu
85 ;*****
86
87 ;;;;;;;;;;;;;;
88 ;; What do these do? Do I need them? -ajh 3/21/96
89
90 (eval-when (:compile-toplevel)
91   (proclaim '(optimize (speed 2)(safety 1)(space 1)(debug 3)))
92   (proclaim '(optimize (speed 3)(safety 0)(space 0)(debug 0)))
93   )
94
95 (eval-when (:compile-toplevel :load-toplevel :execute)
96   (use-package '("UTILITY-FUNCS" "WINDOW-FUNCS" "PPS") "CL-USER"))
97
98
99 ;;;;;;;;;;;;;;
100 ;; Other global variable definitions.
101
102 ;; Control variables
103 (DEFVAR *TRIAL-CLEAR-WINDOWS T)
104 (DEFVAR *STOP-IF-ERROR T)
105
106 ;; Static variables
107 ;; (DEFVAR *STIMULUS NIL) ;holds target object (no longer used? -ajh 7/5/96)
108 (DEFVAR *GO-BOX-OBJECT NIL)
109 (DEFVAR *PRECUE-OBJECT NIL)
110
111 ;; Stimuli variables
112 (DEFVAR *MENU-TEXT-LIST NIL)
113 (DEFVAR *MENU-VISOBJ-LIST NIL)
114 (DEFVAR *MENU-ITEM-COUNTER 0)
115
116 (DEFVAR *CURSOR NIL) ;; holds CURSOR object
117 (DEFVAR *TRACKING-ORIGIN-X 0)
118 (DEFVAR *TRACKING-ORIGIN-Y 0)
119
120 ;; Screen position of go-box and precue. Set below after determining start
121 ;; position of menu and visual angle of menu items.
122 (DEFVAR *GO-BOX-X NIL)
123 (DEFVAR *GO-BOX-Y NIL)
124 (DEFVAR *PRECUE-X NIL)
125 (DEFVAR *PRECUE-Y NIL)
126 (DEFVAR *MENU-START-X 0)
127 (DEFVAR *MENU-START-Y 0)
128 (DEFVAR *TARGET-TEXT NIL) ;; SET BELOW!
129 (DEFVAR *TARGET-OBJECT NIL)
130
131 ;; used for setting the next sweep items in epicfns
132 (DEFVAR *CURRENT-NEXT-OBJ-DOWN NIL)
133 (DEFVAR *CURRENT-NEXT-OBJ-UP NIL)
134 (DEFVAR *NEXT-OBJ-DOWN-CREATED-AT-TIME 0)
135 (DEFVAR *NEXT-OBJ-UP-CREATED-AT-TIME 0)
136 (DEFVAR *NEXT-OBJ-DOWN-CREATED-FOR-MENU-ITEM NIL)
137 (DEFVAR *NEXT-OBJ-UP-CREATED-FOR-MENU-ITEM NIL)
138
139 ;; (DEFVAR *NEXT-SWEEP-LOCATION NIL) ;; no longer used 4/15/01
140 (DEFVAR *CURRENT-GAZE-MENU-ITEM NIL)
141
142 ;; Statistics variables

```

```

143 (DEFVAR *MENU-APPEARS-TIME 0)
144 (DEFVAR *SERIAL-POSITION 0)
145 (DEFVAR *CURRENT-RT 0)
146 (DEFVAR *GRAND-TOT-N 0)
147 (DEFVAR *GRAND-AVG-RT 0)
148 (DEFVAR *GRAND-TOTAL-RT 0)
149 (DEFVAR *TRIAL 0)
150 (DEFVAR *TIMING-IN-PROGRESS NIL) ;; Nil before timing. T during timed trial.
151 (DEFVAR *RESPONSE-TIME 0)
152 (DEFVAR *FIXATION-POSITIONS NIL) ;; Eye movement locations for this trial
153
154 ; cumulative totals
155 (DEFVAR *MAX-NUM-OF-MENU-ITEMS 12)
156 (DEFVAR *CUMULATIVE-RT-FOR-POS (MAKE-ARRAY *MAX-NUM-OF-MENU-ITEMS))
157 (DEFVAR *NUM-TRIALS-FOR-POS (MAKE-ARRAY *MAX-NUM-OF-MENU-ITEMS))
158
159 ;; Eye movement data
160 (defvar *output-eyes-trace) ;; temporary start to adding another output file. 9/20/02
161
162 ;; From -11 to 11, offset by 11
163 ;; Observed: -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12
164 ;; Put in slot: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
165 (DEFVAR *ARRAY-SIZE 24) ;; For arrays holding fixation stats
166 (DEFVAR *FIRST-SACCADE-DISTANCE (MAKE-ARRAY *ARRAY-SIZE))
167 (DEFVAR *MIDDLE-SACCADE-DISTANCE (MAKE-ARRAY *ARRAY-SIZE))
168 (DEFVAR *FINAL-SACCADE-DISTANCE (MAKE-ARRAY *ARRAY-SIZE)) ;; From -11 to 11, offset by 11
169
170 ;; The dimensions of the following two arrays are:
171 ;; Dimension 1 = target location, dimension 2 = menu length
172 (DEFVAR *NUM-FIXATIONS-AS-FUNCTION-OF-TARGET-LOCATION
173 (MAKE-ARRAY (LIST *ARRAY-SIZE *ARRAY-SIZE)))
174 (DEFVAR *FIRST-FIXATION-AS-FUNCTION-OF-TARGET-LOCATION
175 (MAKE-ARRAY (LIST *ARRAY-SIZE *ARRAY-SIZE)))
176
177 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
178 ;; Menu experiment global variables.
179 ;; Cange here or in code to simulate different experimental conditions.
180
181 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
182 (DEFUN INITIALIZE-EXPERIMENT ()
183 ;; Run once per experiment, not for each trial.
184 ;; These need to be reset in case previous block was interrupted.
185 (SETQ *TRIAL 0)
186 (SETQ *TIMING-IN-PROGRESS NIL)
187 ;(SETQ *TARGET-TEXT 1) ;; Initial target is this number + 1
188 ;(SETQ *MENU-TEXT-LIST NIL)
189 (SETQ *MENU-VISOBJ-LIST NIL)
190 (SETQ *MENU-ITEM-COUNTER 0)
191 (SETQ *GO-BOX-OBJECT NIL)
192 (SETQ *PRECUE-OBJECT NIL)
193 (SETQ *TARGET-OBJECT NIL) ;; Would rather reset every trial. -ajh 10/27/96
194 ;; (SETQ *STIMULUS NIL)
195 (SETQ *CURSOR NIL)
196 (SETQ *MENU-START-X 0)
197 (SETQ *MENU-START-Y 3)
198
199 (INITIALIZE-STATS)
200 ;; (INITIALIZE-TRIAL) ;; *** A bunch of stuff is set here. -ajh-5/18/01
201
202 ;; Determine screen position of go-box and precue. Should only have to
203 ;; change if our assumptions of Nilsen's positions are incorrect.
204 ;; (SETQ *GO-BOX-Y (+ *MENU-START-Y *MENU-ITEM-ANGLE)) ;; until 6/6/96
205 ;; (SETQ *GO-BOX-X *MENU-START-X) ;; until 10/27/96 -ajh
206 ;; (SETQ *GO-BOX-X (+ *MENU-START-X (* .1 *MENU-ITEM-ANGLE))) ;; so cursor on target.
207 (SETQ *GO-BOX-X *MENU-START-X) ;; This was just fine, and simpler. 8/16/98 -ajh
208
209 ;; after Nilsen 6/6/96 email, to put cursor just inside first item...
210 (SETQ *GO-BOX-Y (+ *MENU-START-Y (* .45 *MENU-ITEM-ANGLE)))
211 (SETQ *PRECUE-X *MENU-START-X)
212 ;; the 1.5 offset is a guess, but it shouldn't affect outcome provided it is above
213 ;; the go box.

```

```

214 (SETQ *PRECUE-Y (+ *MENU-START-Y (* 1.5 *MENU-ITEM-ANGLE)))
215
216
217
218 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
219 (DEFUN INITIALIZE-STATS ()
220   "Called by INITIALIZE-EXPERIMENT."
221
222   (SETQ *CURRENT-RT 0)
223   (SETQ *GRAND-TOT-N 0)
224   (SETQ *GRAND-AVG-RT 0)
225   (SETQ *GRAND-TOTAL-RT 0)
226
227   (DOTIMES (X *MAX-NUM-OF-MENU-ITEMS)
228     (SETF (AREF *CUMULATIVE-RT-FOR-POS X) 0)
229     (SETF (AREF *NUM-TRIALS-FOR-POS X) 0))
230 )
231
232 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
233 (DEFUN RESET-EYE-MOVEMENT-VARIABLES ()
234   "Called by RUN."
235   (DOTIMES (X *ARRAY-SIZE)
236     (SETF (AREF *FIRST-SACCADE-DISTANCE X) 0)
237     (SETF (AREF *MIDDLE-SACCADE-DISTANCE X) 0)
238     (SETF (AREF *FINAL-SACCADE-DISTANCE X) 0))
239   (DOTIMES (X *ARRAY-SIZE)
240     (DOTIMES (Y *ARRAY-SIZE)
241       (SETF (AREF *NUM-FIXATIONS-AS-FUNCTION-OF-TARGET-LOCATION X Y) 0)))
242   (DOTIMES (X *ARRAY-SIZE)
243     (DOTIMES (Y *ARRAY-SIZE)
244       (SETF (AREF *FIRST-FIXATION-AS-FUNCTION-OF-TARGET-LOCATION X Y) 0)))
245   (SETQ *NUM-MENU-STYLES 0) ;; 1 if one style is run, 2 if both are run.
246   (SETQ *FIXATION-POSITIONS NIL)
247 )
248
249
250
251 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
252 ;; Device high level control, including some input and output
253 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
254
255 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
256 (DEFUN DEV1P (PROCESSOR-START-TIME RESET)
257   (PROG (TIME INPUT-EVENT)
258
259     (COND (RESET
260       (DEV1P-SIGNAL-SELF PROCESSOR-START-TIME 'START-EXPERIMENT)
261       (RETURN NIL)))
262
263     ;TIME keeps track of the time for output events
264     LOOP
265     (SETQ TIME PROCESSOR-START-TIME)
266     (SETQ INPUT-EVENT (RECEIVE-TTD))
267     (COND ((NOT INPUT-EVENT) (RETURN T)))
268     ;do the appropriate thing
269     (CASE (TYPE-OF INPUT-EVENT)
270       (MOVEMENT-EVENT (HANDLE-DEVICE-INPUT TIME INPUT-EVENT))
271       (SIGNAL-EVENT (HANDLE-SIGNAL-EVENT TIME INPUT-EVENT))
272       (T (PRINT (LIST "Unknown input in Dev1:" INPUT-EVENT))
273         (RETURN NIL)))
274     )
275     (GO LOOP)
276   ))
277
278 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
279 (DEFUN HANDLE-SIGNAL-EVENT (TIME INPUT-EVENT)
280   "Weird high-level control mechanism that Dave uses and I'm trying to remove.
281   Only used for DO-START-EXPERIMENT and DO-START-TRIAL
282   Question: Where did these signals come from anyway??? ajh 9/8/95
283   Answer: In statements like (DEV1P-SIGNAL-SELF TIME 'START-TRIAL)
284   that are at the end of the previous functions."

```

```

285
286 (DISPLAY-INPUT TIME "Dev1:"
287 (FORMAT NIL "Signal ~A" (SIGNAL-EVENT-SIGNAL INPUT-EVENT)))
288 (CASE (SIGNAL-EVENT-SIGNAL INPUT-EVENT)
289 (START-EXPERIMENT (DO-START-EXPERIMENT TIME))
290 (START-TRIAL (DO-START-TRIAL TIME))
291 ;; (REMOVE-READY (print (list 'DO-REMOVE-READY 'was 'here)))
292 ;; (DISPLAY-STIMULUS (print (list 'DISPLAY-MENU 'was 'here)))
293 ;; (REMOVE-STIMULUS (print (list 'REMOVE-MENU 'was 'here)))
294 (T (PRINT (LIST "Unknown Signal in Dev1:" INPUT-EVENT)))
295 ))
296
297 ;;;;;;;;;;;;;;
298 ;schedule a message at the specified time to be sent to DEV1P
299 (DEFUN DEV1P-SIGNAL-SELF (TIME SIGNAL)
300 (DISPLAY-OUTPUT TIME "Dev:" SIGNAL)
301 (SEND-TTD TIME (MAKE-SIGNAL-EVENT :SIGNAL SIGNAL) DEV1P)
302 (SCHEDULE-PROCESSOR TIME DEV1P)
303 )
304
305 ;;;;;;;;;;;;;;
306 (DEFUN DO-START-EXPERIMENT (TIME)
307 "Start whole experiment"
308
309 ;create and define the cursor object
310 (SETQ *CURSOR (MAKE-VISUAL-OBJECT-APPEAR TIME
311 *TRACKING-ORIGIN-X *TRACKING-ORIGIN-Y
312 (LIST 'SHAPE 'CROSS 'SIZE 0.5)))
313
314 ;come back after a delay ;; not needed ??? ajh 9/8/95
315 (SETQ TIME (+ TIME 100))
316 (DEV1P-SIGNAL-SELF TIME 'START-TRIAL)
317 )
318
319 ;;;;;;;;;;;;;;
320 (DEFUN DO-START-TRIAL (TIME)
321 "Start each trial.
322 Stop the experiment when all serial positions are used."
323
324 (COND
325 ((NOT *SERIAL-POSITION-LIST)
326 (STOP-EPIC))
327 (T ;; Else if the experiment is not over...
328
329 (SETQ *GO-BOX-OBJECT
330 (MAKE-VISUAL-OBJECT-APPEAR TIME
331 *GO-BOX-X *GO-BOX-Y
332 (LIST 'TEXT 'GO)))
333 (SET-VISUAL-PHYSICAL-PROPERTY *GO-BOX-OBJECT 'IN-MENU 'NO)
334
335 ;; Build menu and determine target item, so precue can be displayed.
336 (BUILD-MENU-CONTENTS)
337
338 (MAKE-PRECUE-APPEAR-IF-APPROPRIATE TIME)
339
340 (UPDATE-OVERVIEW) ;update our overview display only when needed
341
342 ;; Finesse!!! Internal cognition triggered directly in the device file!
343 ;; Represents user's decision about where to make the first
344 ;; saccade when making an efficient sweep down the list. We want to
345 ;; determine it afresh for every trial to build in variation.
346 ;; (Currently only used for randomly-ordered menus. -ajh 4/7/98)
347 (DETERMINE-FIRST-FIXATION-LOCATION *FIRST-FIXATION)
348
349 ;; Finesse! Again, the device driver reaches into the simulated human.
350 ;; Rather than creating a new movement location (with variable error)
351 ;; to all possible locations (for ordered menus), just create one
352 ;; movement to the known target location. Just to save programming cycles.
353 (ADD-ERROR-TO-NAMED-LOCATION-OF-TARGET )
354
355 ))

```

```

356 )
357
358 ;;;;;;;;;;;;;;
359 (DEFUN DO-REMOVE-READY (TIME)
360 ;Remove GO-BOX and precue
361
362 (MAKE-VISUAL-OBJECT-DISAPPEAR TIME *GO-BOX-OBJECT)
363 (MAKE-VISUAL-OBJECT-DISAPPEAR TIME *PRECUE-OBJECT) ;; should be when cursor moves
364
365 (SETQ *GO-BOX-OBJECT NIL)
366 (SETQ *PRECUE-OBJECT NIL)
367 (UPDATE-OVERVIEW)
368
369 )
370
371
372 ;;;;;;;;;;;;;;
373 ;; Input to device, lower level functions
374 ;;;;;;;;;;;;;;
375
376 (DEFUN HANDLE-DEVICE-INPUT (TIME INPUT-EVENT)
377 (LET ()
378 (DISPLAY-INPUT TIME "Dev1:" (STRING-OF-MOVEMENT-EVENT INPUT-EVENT))
379
380 (CASE (MOVEMENT-EVENT-DEVICE-INPUT-TYPE INPUT-EVENT)
381 (MOUSE
382 (DO-MOUSE-MOVEMENT TIME INPUT-EVENT)
383 )
384 (MOUSE-BUTTON
385 (DO-MOUSE-CLICK TIME)
386 (if *output-eyes-trace
387 (print (list 'mouse 'clicked 'at 'time TIME)))
388 ))
389 ))
390
391
392 ;;;;;;;;;;;;;;
393 (DEFUN DO-MOUSE-CLICK (TIME)
394 ;; If the cursor is in the "go" box location and the timing has not yet
395 ;; started, then put up the menu and start the timing.
396 ;; If the timing is already underway, then stop the timing. Report
397 ;; to the subject and to the statistics record whether the response
398 ;; was correct or not. -ajh 9/8/95
399 (LET ()
400
401 (COND
402
403 ;; Subject initiates
404 ((NOT *TIMING-IN-PROGRESS)
405 (SETQ *TIMING-IN-PROGRESS T)
406 (DO-REMOVE-READY TIME)
407 (DISPLAY-MENU TIME)
408 ;save the current time when the menu appeared
409 (SETQ *MENU-APPEARS-TIME TIME)
410 ;;trash this: (print (list 'timing 'starts 'at *MENU-APPEARS-TIME))
411 ;the next event should be the response, a movement event
412 )
413
414 ;; Subject responds (*TIMING-IN-PROGRESS is true)
415 (T
416
417 (VERIFY-CLICK-ON-TARGET)
418
419
420 (SETQ *TIMING-IN-PROGRESS NIL)
421 (DISPLAY-INPUT TIME "Dev:"
422 (FORMAT T "~%Response is recorded."))
423
424 (DO-TRIAL-STATS *MENU-APPEARS-TIME TIME)
425
426 ;; (SEND-TTD TIME NIL VSENP) ;; removed 7/2/96 -ajh

```

```

427 ;; (SCHEDULE-PROCESSOR TIME VSENP)
428 (SETQ *TRIAL (1+ *TRIAL))
429
430 (REMOVE-MENU (+ TIME 100))
431 (DO-START-TRIAL (+ TIME 2000))
432
433 ;; Perhaps we only want to do these when we are done with all trials.
434 ;; Perhaps they need separate function. ajh 9/8/95
435 ;; (TRIAL-JITTER) ;randomize model parameters if option set
436 ;; (IF *TRIAL-CLEAR-WINDOWS (CLEAREM))
437 ))
438 ))
439
440 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
441 (DEFUN VERIFY-CLICK-ON-TARGET ()
442 "Determine whether or not the subject clicks on the target item. -ajh -4/7/98"
443 (LET (Y-START Y-TARGET Y-LAND Y-CORRECT-DISTANCE Y-ACTUAL-DISTANCE Y-ERROR)
444
445 (COND
446
447 (*OUTPUT-POINT-ERRORS
448
449 ;; perhaps should be disabled because does not work when there is more
450 ;; than one POINT in the strategy
451
452 (SETQ Y-START *GO-BOX-Y)
453
454 ;; Determine the target for the POINT or SACCADE.
455 (SETQ Y-TARGET (LOCATION-Y (GET-VISUAL-OBJECT-PHYSICAL-LOCATION *TARGET-OBJECT)))
456
457 ;; Where the POINT or SACCADE actually landed.
458 (SETQ Y-LAND (LOCATION-Y (GET-VISUAL-OBJECT-PHYSICAL-LOCATION *CURSOR)))
459
460 (SETQ Y-CORRECT-DISTANCE (ABS (- Y-START Y-TARGET)))
461 (SETQ Y-ACTUAL-DISTANCE (ABS (- Y-START Y-LAND)))
462
463 ;; Determine vertical error between target and where POINT or SACCADE landed.
464 (SETQ Y-ERROR (- Y-ACTUAL-DISTANCE Y-CORRECT-DISTANCE))
465
466
467 (DOLIST (OUTPUT-STREAM '(*STANDARD-OUTPUT* *RESULTS-FILE))
468 ;; Output Target digit, target position, start position, and error.
469 (PRINTC (FORMAT NIL "~A~A~,6F~A~,6F~A~,6F"
470 *TARGET-TEXT #\tab Y-TARGET #\tab *GO-BOX-Y #\tab Y-ERROR) (EVAL
... OUTPUT-STREAM))
471 ;;(print OUTPUT-STREAM)
472 )
473 )
474
475 ;; ...else if you are not displaying POINT errors...
476 (T
477 ;; If cursor is not on target item, break with an error.
478 (IF (NOT (CURSOR-ON-OBJECT-P *CURSOR *TARGET-OBJECT *MENU-ITEM-ANGLE))
479 (IF *STOP-IF-ERROR
480 (BREAK (FORMAT NIL
481 "SUBJECT MISSED the target ~A with the cursor ~A"
482 *TARGET-OBJECT *CURSOR))))))
483
484 ))
485
486 ;; Perhaps discard.....
487 ;; If cursor is not on target item, issue an error and continue.
488 ;;(IF (NOT (CURSOR-ON-OBJECT-P *CURSOR *TARGET-OBJECT *MENU-ITEM-ANGLE))
489 ;; (PRINTC (FORMAT NIL
490 "SUBJECT MISSED the target ~A with the cursor ~A"
491 *TARGET-OBJECT *CURSOR) *STANDARD-OUTPUT*))
492
493
494 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
495 (DEFUN DO-MOUSE-MOVEMENT (TIME INPUT-EVENT)
496 ;; Move the crosshairs as specified by the input event of the mouse movement.

```

```

497 ;; This function gets executed once for each mouse/cursor movement.
498 ;; TIME is the time that the mouse movement is completed, and stays the same
499 ;; throughout the execution of this function.
500
501 (LET (CHANGE-X CHANGE-Y NEW-X NEW-Y OUTPUT POINTS-TO-TMP)
502
503   ;; Get the movement and convert it to change in X and Y.
504   ;; (I believe the X and Y values themselves are also available. -ajh 5/19/98)
505   (MULTIPLE-VALUE-SETQ (CHANGE-X CHANGE-Y)
506     (POLAR-TO-CARTESIAN
507       (POLAR-VECTOR-R (MOVEMENT-EVENT-VECTOR INPUT-EVENT))
508       (POLAR-VECTOR-THETA (MOVEMENT-EVENT-VECTOR INPUT-EVENT))))
509
510   (SETQ NEW-X
511     (+ CHANGE-X (LOCATION-X (GET-VISUAL-OBJECT-PHYSICAL-LOCATION *CURSOR))))
512   (SETQ NEW-Y
513     (+ CHANGE-Y (LOCATION-Y (GET-VISUAL-OBJECT-PHYSICAL-LOCATION *CURSOR))))
514
515   (SET-VISUAL-OBJECT-PHYSICAL-LOCATION *CURSOR (MAKE-LOCATION :X NEW-X :Y NEW-Y))
516
517   ;; Set CURSOR POINTS-TO to the menu item pointed to, or to nothing if it points
518   ;; to no item. Set it to NIL to remove it if the trial is not in progress.
519   ;; -ajh 5/19/98
520   (SETQ POINTS-TO-TMP 'NOTHING)
521   (COND
522     (*TIMING-IN-PROGRESS
523      (DOLIST (OBJ *MENU-VISOBJ-LIST)
524        (COND ((CURSOR-ON-OBJECT-P *CURSOR OBJ *MENU-ITEM-ANGLE)
525              (SETQ POINTS-TO-TMP OBJ)))))
526     (T
527      (SETQ POINTS-TO-TMP NIL)))
528
529   ;; Queue the event to update the POINTS-TO feature -ajh 5/22/98
530   (SET-VISUAL-PHYSICAL-PROPERTY *CURSOR 'POINTS-TO POINTS-TO-TMP)
531   (SETQ OUTPUT (MAKE-VISUAL-EVENT :OBJECT *CURSOR :CHANGE 'POINTS-TO))
532   (DISPLAY-OUTPUT TIME "Dev1" (FORMAT NIL "~A Points to ~A" *CURSOR POINTS-TO-TMP))
533   (SEND-TTD TIME OUTPUT VSENP)
534   (SCHEDULE-PROCESSOR TIME VSENP)
535   (UPDATE-OVERVIEW)
536
537   ;; Queue the event to update the location.
538   ;; Copied from Wickens. Do something similar.
539   ;note - any queued events for this signal will be pointing
540   ; to the CURRENT version of *CURSOR's properties
541   (SETQ OUTPUT (MAKE-VISUAL-EVENT :OBJECT *CURSOR :CHANGE 'LOCATION))
542   (DISPLAY-OUTPUT TIME "Dev1" (FORMAT NIL "~A Moving to ~A" *CURSOR
543     (STRING-OF-LOCATION (GET-VISUAL-OBJECT-PHYSICAL-LOCATION *CURSOR))))
544   (SEND-TTD TIME OUTPUT VSENP)
545   (SCHEDULE-PROCESSOR TIME VSENP)
546   (UPDATE-OVERVIEW)
547
548   (MAKE-PRECUE-APPEAR-IF-APPROPRIATE (+ TIME 100)) ;; delay 11/7/97 ajh
549
550 ))
551
552
553 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
554 (DEFUN MAKE-PRECUE-APPEAR-IF-APPROPRIATE (TIME)
555   "If during precue stage, the go box has appeared, the mouse is in go box,
556   and the precue has not yet appeared, then make precue (to-be-selected item)
557   appear in the visual space. ajh."
558
559   (COND
560     ((AND
561       (NOT *TIMING-IN-PROGRESS)
562       *GO-BOX-OBJECT
563       (CURSOR-ON-OBJECT-P *CURSOR *GO-BOX-OBJECT *MENU-ITEM-ANGLE)
564       (NOT *PRECUE-OBJECT))
565
566      (SETQ *PRECUE-OBJECT
567        (MAKE-VISUAL-OBJECT-APPEAR

```

```

568         TIME *PRECUE-X *PRECUE-Y
569
570         ;; Uncomment one of the next two lines.
571
572         ;; For debugging, so EPIC will search forever -ajh 4/15/01
573         ;; (LIST 'TEXT 'X))
574
575         ;; For normal operation, so EPIC will find the target.
576         (LIST 'TEXT *TARGET-TEXT))
577
578         ;; Give the precue a feature to distinguish it from the actual menu item.
579         ;; These do not seem to use the delays from epicfns file.
580         (SET-VISUAL-PHYSICAL-PROPERTY *PRECUE-OBJECT 'IN-MENU 'NO)
581         (SET-VISUAL-PHYSICAL-PROPERTY *PRECUE-OBJECT 'IS-ABOVE *GO-BOX-OBJECT)
582
583         (UPDATE-OVERVIEW) ;update our overview display only when needed
584     ))
585 )
586
587
588 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
589 (DEFUN CURSOR-ON-OBJECT-P (CURSOR TARGET-OBJ ITEM-HEIGHT)
590     "Determine if the cursor is currently positioned on a specified object.
591     An object's location is thought to be its center.
592     The object (as of 10/27/96) is either the GO BOX or a target menu item.
593     Since no item width is set, just use the height (as if target is square)."

```

```

639 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
640
641 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
642 (DEFUN BUILD-MENU-CONTENTS ()
643   "Build the text contents of the menu. Store the text in *MENU-TEXT-LIST,
644   which will later be transformed into *MENU-VISOBJ-LIST."
645
646   ;; Create the list of menu items.
647   (SETQ *MENU-TEXT-LIST (DIGIT-LIST *MENU-ORDERED *MENU-NUMBER-ITEMS))
648
649   ;; Get the next desired serial position from the list, set it as the target.
650   (SETQ *SERIAL-POSITION (CAR *SERIAL-POSITION-LIST))
651   (SETQ *SERIAL-POSITION-LIST (CDR *SERIAL-POSITION-LIST))
652   (SETQ *TARGET-TEXT (NTH (- *SERIAL-POSITION 1) *MENU-TEXT-LIST))
653 )
654
655 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
656 (DEFUN DISPLAY-MENU (TIME)
657   ;; Build and display the EPIC physical objects for the textual menu list.
658
659   ;; Determine the correct serial position.
660   (SETQ *SERIAL-POSITION (+ 1 (position *TARGET-TEXT *MENU-TEXT-LIST)))
661
662   ;; Reset the menu item list here for lack of an obvious place higher up.
663   (SETQ *MENU-VISOBJ-LIST NIL)
664   (SETQ *MENU-ITEM-COUNTER 0)
665
666   (BUILD-MENU-OBJECTS *MENU-TEXT-LIST 'vert *MENU-ITEM-ANGLE
667     (LIST *MENU-START-X *MENU-START-Y) TIME)
668   (UPDATE-OVERVIEW)
669 )
670
671 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
672 (DEFUN BUILD-MENU-OBJECTS (ITEM-LIST HORIZ-VERT ANGLE-SEP START-POINT TIME)
673   "Given a list of text objects, a horizontal versus vertical orientation,
674   an angle of separation between each consecutive pair of text objects, and a
675   starting point in the visual space, create a physical menu list of these
676   objects in the visual space and link them with the appropriate features.
677   Note: The initial version of the function will not wrap lists."
678
679   (LET (NEW-ITEM ITEM-NUMBER X-POS Y-POS)
680     (SETQ ITEM-NUMBER 0)
681     (SETQ X-POS (CAR START-POINT))
682     (SETQ Y-POS (CADR START-POINT))
683
684     (IF (< (LENGTH ITEM-LIST) 1)
685       (IF *STOP-IF-ERROR
686         (BREAK "BUILD-MENU-OBJECTS requires at least one item.")))
687
688     ;; Create each menu item and its required properties.
689     (DOLIST (ITEM ITEM-LIST)
690       (SETQ NEW-ITEM
691         (MAKE-VISUAL-OBJECT-APPEAR      ;; Create the object.
692          TIME
693          X-POS Y-POS
694          (LIST 'IN-MENU 'YES
695               'COLOR 'BLK
696               'TEXT ITEM
697               'SIZE ANGLE-SEP ;; in this case menu height is Fitts width
698               )))
699
700     ;; If this is the target item...
701     (COND
702       ((EQUAL ITEM *TARGET-TEXT)
703        ;; Make a special pointer to the object. -ajh 10/27/96
704        (SETQ *TARGET-OBJECT NEW-ITEM)
705       ))
706
707     ;; Pass through the physical boundaries (vertical only) of the object.
708     (SET-VISUAL-PHYSICAL-PROPERTY NEW-ITEM 'TOP-BOUNDARY (+ Y-POS (/ ANGLE-SEP 2)))
709     (SET-VISUAL-PHYSICAL-PROPERTY NEW-ITEM 'BOTTOM-BOUNDARY (- Y-POS (/ ANGLE-SEP 2)))

```

```

710
711 (SET-VISUAL-PHYSICAL-PROPERTY NEW-ITEM 'SERIAL-POSITION (+ 1 ITEM-NUMBER))
712
713 ;; Increment loop variables.
714 (IF (EQUAL HORIZ-VERT 'VERT) (SETQ Y-POS (- Y-POS ANGLE-SEP)))
715 (IF (EQUAL HORIZ-VERT 'HORIZ) (SETQ X-POS (+ X-POS ANGLE-SEP)))
716 (INCF ITEM-NUMBER)
717 (SETQ *MENU-VISOBJ-LIST (CONS NEW-ITEM *MENU-VISOBJ-LIST))
718 (INCF *MENU-ITEM-COUNTER)
719 )
720
721 ;; Assign the properties that are specific to vertical or horiz. menu.
722 (COND
723
724   ;; Vertical menu.
725   ((EQUAL HORIZ-VERT 'VERT)
726    ;; Step through the objects and assign IS-ABOVE's and IS-BELOW's.
727    (DOTIMES (ITEM-NUM (- *MENU-ITEM-COUNTER 1))
728
729      (SET-VISUAL-PHYSICAL-PROPERTY (NTH ITEM-NUM *MENU-VISOBJ-LIST)
730      'IS-BELOW (NTH (+ 1 ITEM-NUM) *MENU-VISOBJ-LIST))
731      (SET-VISUAL-PHYSICAL-PROPERTY (NTH (+ 1 ITEM-NUM) *MENU-VISOBJ-LIST)
732      'IS-ABOVE (NTH ITEM-NUM *MENU-VISOBJ-LIST))
733      ) ;; End of DOTIMES assigning above and below.
734
735      (SET-VISUAL-PHYSICAL-PROPERTY (CAR *MENU-VISOBJ-LIST)
736      'IS-ABOVE 'NOTHING)
737      (SET-VISUAL-PHYSICAL-PROPERTY (CAR (LAST *MENU-VISOBJ-LIST))
738      'IS-BELOW 'NOTHING)
739
740      (SET-VISUAL-PHYSICAL-PROPERTY (CAR *MENU-VISOBJ-LIST)
741      'POSITION-IN-MENU 'LAST)
742      (SET-VISUAL-PHYSICAL-PROPERTY (CAR (LAST *MENU-VISOBJ-LIST))
743      'POSITION-IN-MENU 'FIRST))
744
745      ((EQUAL HORIZ-VERT 'HORIZ)
746       (IF *STOP-IF-ERROR (BREAK "HORIZ menu not implemented yet.")))
747
748      (T (IF *STOP-IF-ERROR (BREAK "BUILD-MENU-OBJECTS requires HORIZ or VERT."))))
749 ))
750
751 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
752 (DEFUN DIGIT-LIST (ORDERED-P NUM-DIGITS)
753   "Make a list of the digits from 1 to NUM-DIGITS. If ORDERED-P is NIL,
754   randomize the list. Return the list."
755   (LET (INIT-LIST THIS-ITEM NEW-LIST)
756
757     ;; Build an ordered list of digits from 1 to NUM-DIGITS.
758     (DOTIMES (X NUM-DIGITS)
759      (SETQ INIT-LIST (APPEND INIT-LIST (LIST (+ 1 X)))))
760
761     (COND
762      ((NOT ORDERED-P) ;; If random
763       (DOTIMES (X (LENGTH INIT-LIST))
764        ;; Pick a random number from 1 to the number of items left in the set.
765        (SETQ THIS-ITEM (NTH (RANDOM (LENGTH INIT-LIST)) INIT-LIST))
766        (SETQ NEW-LIST (CONS THIS-ITEM NEW-LIST))
767        ;; Pull out that item. (Note: DELETE is destructive; REMOVE is not.)
768        (SETQ INIT-LIST (REMOVE THIS-ITEM INIT-LIST)))
769       )
770      (T (SETQ NEW-LIST INIT-LIST))) ;; Set the var to return if ordered.
771
772     NEW-LIST
773   ))
774
775
776
777 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
778 (DEFUN MAKE-SERIAL-POSITION-LIST (NUM-ITEMS TRIALS-PER-POS)
779   "Generate a random list of numbers ranging from 1 to NUM-ITEMS, such that
780   each number occurs exactly TRIALS-PER-POS times. Create an ordered list,

```

```

781 then randomly select and delete one item at a time, putting each into a
782 new list pointed to by *SERIAL-POSITION-LIST.
783
784 Dave's idea: Create an ordered list within an array.  Swapping each array
785 item with another randomly chosen array item."
786
787 (LET (ORDERED-LIST NEXT-ITEM)
788
789 (SETQ *SERIAL-POSITION-LIST NIL)
790
791 ;; Start the list (so we can use NCONC)
792 (SETQ ORDERED-LIST (MAKE-LIST TRIALS-PER-POS :INITIAL-ELEMENT '1))
793
794 ;; Create an ordered list.
795 (DOTIMES (X (- NUM-ITEMS 1))
796   (NCONC ORDERED-LIST (MAKE-LIST TRIALS-PER-POS :INITIAL-ELEMENT (+ 2 X)))
797 )
798
799 ;; Until the ordered list is empty...
800 (DOTIMES (X (LENGTH ORDERED-LIST))
801
802   ;; Select a random item from the ordered list.
803   (SETQ NEXT-ITEM (NTH (RANDOM (LENGTH ORDERED-LIST)) ORDERED-LIST))
804
805   ;; Put it in the random list
806   (PUSH NEXT-ITEM *SERIAL-POSITION-LIST)
807
808   ;; Remove it (or one just like it) from the ordered list.
809   ;; (DELETE didn't work here.)
810   (SETQ ORDERED-LIST (REMOVE NEXT-ITEM ORDERED-LIST :COUNT 1))
811 )
812 ))
813
814
815 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
816 (DEFUN DISPLAY-BAD-INPUT (TIME RESPONSE)
817   (DISPLAY-INPUT TIME "Dev:"
818     (FORMAT NIL "Response ~A received when not expected" RESPONSE))
819   (IF *STOP-IF-ERROR (BREAK "Bogus Response!")))
820
821
822 (DEFUN REMOVE-MENU (TIME)
823   (LET ()
824     ;; clear the menu from the visual space
825     (DOLIST (ITEM *MENU-VISOBJ-LIST)
826       (MAKE-VISUAL-OBJECT-DISAPPEAR TIME ITEM))
827
828     (UPDATE-OVERVIEW)
829   ))
830
831
832 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
833 ;; Keep track of experimental results.
834 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
835
836 (DEFUN DO-TRIAL-STATS (MENU-APPEARED-TIME RESPONSE-TIME)
837   (LET (RESULT)
838     (SETQ *CURRENT-RT (- RESPONSE-TIME MENU-APPEARED-TIME))
839     ;update grand averages data for display purposes
840     (SETQ *GRAND-TOT-N (1+ *GRAND-TOT-N))
841     (SETQ *GRAND-TOTAL-RT (+ *GRAND-TOTAL-RT *CURRENT-RT))
842     (SETQ *GRAND-AVG-RT (/ *GRAND-TOTAL-RT *GRAND-TOT-N))
843
844     ;update totals based on serial position
845
846     (INCF (AREF *CUMULATIVE-RT-FOR-POS (- *SERIAL-POSITION 1)) *CURRENT-RT)
847     (INCF (AREF *NUM-TRIALS-FOR-POS (- *SERIAL-POSITION 1)))
848
849     (DO-EYE-MOVEMENT-TRIAL-STATS)
850
851     (COND (*OUTPUT-RT-DATA-EVERY-TRIAL

```

```

852         ;(SETQ RESULT (FORMAT NIL " ~4D, ~12D, ~16D"
853         ;           (+ 1 *TRIAL) *SERIAL-POSITION *CURRENT-RT))
854     (SETQ RESULT (FORMAT NIL "~5D~A~12D~A~14D"
855         (+ 1 *TRIAL) #\tab *SERIAL-POSITION #\tab *CURRENT-RT))
856     (PRINC RESULT *RESULTS-FILE)
857     (PRINC RESULT *STANDARD-OUTPUT*)
858 ))
859
860 (DISPLAY-INPUT RESPONSE-TIME "Dev:" RESULT)
861 ))
862
863
864 ;;;;;;;;;;;;;;
865 (DEFUN DO-EYE-MOVEMENT-TRIAL-STATS ()
866     "Collect the eye movement data necessary to compare against Byrne's IJHCS article
867     and Nilsen's CHI 98 PowerPoint slides.
868     Note that *FIXATION-POSITIONS are actually stored in reverse-order.
869     In retrospect, I should have just used a REVERSE and put them in the right order.
870     ajh-6/10/01"
871
872     ;; Collect the number of fixations as a function of the target position.
873     (INCF (AREF *NUM-FIXATIONS-AS-FUNCTION-OF-TARGET-LOCATION
874         *SERIAL-POSITION *MENU-NUMBER-ITEMS) (LENGTH *FIXATION-POSITIONS))
875
876     ;; Collect the first fixation as a function of the target position.
877     ;; Add up the distance to the first fixation for each target position and menu length
878     (INCF (AREF *FIRST-FIXATION-AS-FUNCTION-OF-TARGET-LOCATION
879         *SERIAL-POSITION *MENU-NUMBER-ITEMS) (CAR (LAST *FIXATION-POSITIONS)))
880
881     ; (INCF (AREF *FIRST-FIXATION-AS-FUNCTION-OF-TARGET-LOCATION
882     ;     (CAR (LAST *FIXATION-POSITIONS)) *MENU-NUMBER-ITEMS))
883
884     ;; Collect the first, middle, and last saccade frequencies. Note that if there
885     ;; is only one saccade, it will get counted as both the first AND the last saccade.
886     ;; Also note the offset by 11 in the middle and final saccade arrays.
887
888     (IF (> *MENU-NUMBER-ITEMS 3)
889
890         ;; If there are more than 3 items in the menu....
891         ;; then pop and classify each fixation.
892         (COND
893             ((EQL 0 (LENGTH *FIXATION-POSITIONS))
894              (BREAK (FORMAT NIL "No *FIXATION-POSITIONS recorded.")))
895             ((EQL 1 (LENGTH *FIXATION-POSITIONS))
896              (INCF (AREF *FINAL-SACCADE-DISTANCE (+ 11 (CAR *FIXATION-POSITIONS))))
897              (INCF (AREF *FIRST-SACCADE-DISTANCE (+ 11 (POP *FIXATION-POSITIONS)))))
898             ((EQL 2 (LENGTH *FIXATION-POSITIONS))
899              (INCF (AREF *FINAL-SACCADE-DISTANCE
900                  (+ 11 (POP-AND-DETERMINE-DISTANCE))))
901              (INCF (AREF *FIRST-SACCADE-DISTANCE (+ 11 (POP *FIXATION-POSITIONS)))))
902             (T ;; There are 3 or more fixation positions.
903              (INCF (AREF *FINAL-SACCADE-DISTANCE
904                  (+ 11 (POP-AND-DETERMINE-DISTANCE))))
905              (LOOP WHILE (> (LENGTH *FIXATION-POSITIONS) 1) DO
906                  (INCF (AREF *MIDDLE-SACCADE-DISTANCE
907                      (+ 11 (POP-AND-DETERMINE-DISTANCE)))))
908              (INCF (AREF *FIRST-SACCADE-DISTANCE (+ 11 (POP *FIXATION-POSITIONS)))))
909
910         ;; Else there are 3 or fewer items in the menu....
911         (SETQ *FIXATION-POSITIONS NIL))
912
913     (IF *FIXATION-POSITIONS
914         (BREAK (FORMAT NIL "*FIXATION-POSITIONS ~A was not fully emptied."
915             *FIXATION-POSITIONS)))
916 )
917
918 ;;;;;;;;;;;;;;
919 (DEFUN POP-AND-DETERMINE-DISTANCE ()
920     "Given a *FIXATION-POSITIONS list with 2 or more elements, pop the first element
921     on the list (which is actually in reverse order) and return the distance between
922     the first two elements on the list. (5 3) => 2 (3 5) => -2"

```

[illegible]

```

979             (CALC-FREQUENCY-RESULT *MIDDLE-SACCADE-DISTANCE X MSD-TOTAL)
980             #\tab
981             (CALC-FREQUENCY-RESULT *FINAL-SACCADE-DISTANCE X FLSD-TOTAL)
982             )))
983         (PRINC RESULT OUTPUT-STREAM))
984
985     ))
986
987
988     ;;;;;;;;;;;;;;
989     (DEFUN SUM-ARRAY (THE-ARRAY)
990       "Return the sum of all the elements in the array."
991       (LET ((RETURN-VALUE 0))
992         (DOTIMES (X *ARRAY-SIZE)
993           (INCF RETURN-VALUE (AREF THE-ARRAY X)))
994         RETURN-VALUE)
995     ))
996
997
998     ;;;;;;;;;;;;;;
999     (DEFUN CALC-FIXATION-RESULT (THE-ARRAY POS MENU-LENGTH)
1000     "Given the array holding the number of fixations to each location across
1001     all trials, a position number, and the menu length; return the
1002     average number of fixations to that location in each trial, or a period
1003     if there is no result for that position. (Returning a space confused
1004     CricketGraph.)
1005
1006     Given the array holding the total distance from the GO box to the
1007     target item across all trials, indexed by target position and menu length,
1008     return the average location (position) of the first fixation. ajh-6/14/01"
1009
1010     (IF (< POS MENU-LENGTH)
1011       (float
1012         (/ (AREF THE-ARRAY (+ 1 POS) MENU-LENGTH)
1013           (* *TOTAL-TRIALS-PER-POSITION *NUM-MENU-STYLES)))
1014       #\.)
1015     )
1016
1017
1018     ;;;;;;;;;;;;;;
1019     (DEFUN CALC-FREQUENCY-RESULT (THE-ARRAY POS TOTAL)
1020     "Given the array holding the total number of occurrences of a saccade distance,
1021     indexed by distance; an index into the array; and the total number of
1022     distances recorded; return the frequency with which a saccade of that
1023     distance occurred ajh-6/14/01"
1024
1025     ;; Don't print the results for a distance of 0, which is indexed by 11
1026     ;; because the array starts at -11
1027
1028     (COND ((EQL 0 TOTAL) 0) ;; Just return 0 if TOTAL is zero
1029           ((< POS 11) (FLOAT (/ (AREF THE-ARRAY POS) TOTAL)))
1030           (T (FLOAT (/ (AREF THE-ARRAY (+ 1 POS)) TOTAL))))
1031     )
1032
1033
1034     ;;;;;;;;;;;;;;
1035     (DEFUN DO-FINAL-STATS (&OPTIONAL (OUTPUT-STREAM NIL))
1036       (LET (RESULT)
1037         ;eventually use the OUTPUT-STREAM in the PRINC's
1038         ;output cumulative results
1039
1040         (TERPRI OUTPUT-STREAM)
1041         (PRINC (FORMAT NIL "-----")
1042           OUTPUT-STREAM)
1043         (PRINC (FORMAT NIL "Serial Position      Number Trials      Avg Select Time")
1044           OUTPUT-STREAM)
1045         (PRINC (FORMAT NIL "-----")
1046           OUTPUT-STREAM)
1047
1048         ; Print cumulative results.
1049

```

```

1050 (DOTIMES (X *MENU-NUMBER-ITEMS)
1051 (SETQ RESULT (FORMAT NIL "~8D~A~11D~A~15D"
1052 (+ 1 X) #\tab (AREF *NUM-TRIALS-FOR-POS X) #\tab
1053 (ROUND (/ (AREF *CUMULATIVE-RT-FOR-POS X)
1054 (AREF *NUM-TRIALS-FOR-POS X)))))
1055 (PRINC RESULT OUTPUT-STREAM))
1056
1057 (PRINC (FORMAT NIL "-----")
1058 OUTPUT-STREAM)
1059
1060 (SETQ RESULT (FORMAT NIL "
1061 Averages: ~5D~A~15D"
1062 (ROUND (/ *GRAND-TOT-N *MENU-NUMBER-ITEMS)) #\tab
1063 (ROUND *GRAND-AVG-RT)))
1064 (PRINC RESULT OUTPUT-STREAM)
1065 ; (DISPLAY-INPUT *MASTER-CLOCK "Dev:" RESULT)
1066 ; (PRINC RESULT *RESULTS-FILE)
1067 ))
1068
1069
1070
1071 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1072 ;; Refresh the EPIC visual space.
1073 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1074
1075 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1076 (DEFUN UPDATE-OVERVIEW ()
1077 "shadows the one in overview-display.lisp
1078 clear and rewrite the overview window to show the current external
1079 situation. Shadow this with an application specific function as needed
1080 NOTE CALL TO GRAPHIC OVERVIEW UPDATE FUNCTION"
1081
1082 (UPDATE-GRAPHIC-OVERVIEW)
1083
1084 (IF (NOT *OVERVIEW-WINDOW)
1085 (RETURN-FROM UPDATE-OVERVIEW NIL))
1086 (WINDOW-CLEAR *OVERVIEW-WINDOW)
1087 (WINDOW-PRINC *MASTER-CLOCK *OVERVIEW-WINDOW)
1088 (WINDOW-PRINC
1089 (LIST "Eye at: " (STRING-OF-LOCATION (GET-EYE-LOCATION)))
1090 *OVERVIEW-WINDOW)
1091 (DOLIST (OBJECT *VISUAL-PHYSICAL-SPACE)
1092 (WINDOW-PRINC
1093 (LIST (GET-VISUAL-PHYSICAL-PROPERTY OBJECT 'SHAPE) (GET-VISUAL-PHYSICAL-PROPERTY
1094 ... OBJECT 'TEXT)
1095 "at"
1096 (STRING-OF-LOCATION (GET-VISUAL-OBJECT-PHYSICAL-LOCATION OBJECT)))
1097 *OVERVIEW-WINDOW)
1098 )
1099 (WINDOW-PRINC (FORMAT NIL "Trial ~D: RT: ~6,0F" *TRIAL *CURRENT-RT) *OVERVIEW-WINDOW)
1100 ;force last line to show up
1101 (WINDOW-PRINC "----" *OVERVIEW-WINDOW)
1102 (WINDOW-TERPRI *OVERVIEW-WINDOW)
1103 )
1104 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1105 (DEFUN UPDATE-GRAPHIC-OVERVIEW ()
1106 "shadows the one in overview-display.lisp
1107 clear and rewrite the overview window to show the current external
1108 situation. Shadow this with an application specific function as needed"
1109
1110 (LET (TEXT SHAPE X Y SIZE COLOR MANUAL-STYLE MANUAL-HAND MANUAL-FINGER
1111 MANUAL-FEATURES MANUAL-ACTION OCULOMOTOR-ACTION)
1112 (IF (NOT *GRAPHIC-OVERVIEW-WINDOW)
1113 (RETURN-FROM UPDATE-GRAPHIC-OVERVIEW NIL))
1114 (ERASE-GRAPHIC-WINDOW *GRAPHIC-OVERVIEW-WINDOW)
1115 (SETQ MANUAL-ACTION (GET-PROCESSOR STATE MOVEMENT-IN-PROGRESS MMOTP))
1116 (COND (MANUAL-ACTION
1117 (SETQ MANUAL-FEATURES (GET-PROCESSOR STATE EXECUTION-MOVEMENT-FEATURES MMOTP))
1118 (SETQ MANUAL-STYLE (GETF MANUAL-FEATURES 'STYLE))
1119 (SETQ MANUAL-HAND (GETF MANUAL-FEATURES 'HAND))

```

```

1120      (SETQ MANUAL-FINGER (GETF MANUAL-FEATURES 'FINGER)))
1121 (SETQ OCULOMOTOR-ACTION (GET-PROCESSOR STATE MOVEMENT-IN-PROGRESS OMOTP))
1122 (DRAW-VISUAL-OBJECT 'ABSOLUTE-TEXT
1123      0 10
1124      (FORMAT NIL "Time:~D" *MASTER-CLOCK ))
1125 (DRAW-VISUAL-OBJECT 'ABSOLUTE-TEXT
1126      0 20
1127      (FORMAT NIL "Manual:~A ~A ~A, Ocular:~A"
1128              MANUAL-STYLE MANUAL-HAND MANUAL-FINGER OCULOMOTOR-ACTION))
1129 (DRAW-VISUAL-OBJECT 'ABSOLUTE-TEXT
1130      0 30
1131      (FORMAT NIL "Target:~A" *TARGET-TEXT))
1132
1133 (DRAW-VISUAL-OBJECT 'ABSOLUTE-TEXT
1134      0 (- *GRAPHIC-OVERVIEW-WINDOW-VSIZE 10)
1135      (FORMAT NIL "Trial:~D RT:~5,0F Avg. RT:~5,0F" *TRIAL *CURRENT-RT
... *GRAND-AVG-RT))
1136
1137
1138 ;draw the visual state
1139 (DRAW-EYE)
1140 ;draw each object in physical visual space
1141 (DOLIST (OBJECT *VISUAL-PHYSICAL-SPACE)
1142      (SETQ TEXT (GET-VISUAL-PHYSICAL-PROPERTY OBJECT 'TEXT))
1143      (SETQ SHAPE (GET-VISUAL-PHYSICAL-PROPERTY OBJECT 'SHAPE))
1144      (SETQ SIZE (GET-VISUAL-PHYSICAL-PROPERTY OBJECT 'SIZE))
1145      (SETQ COLOR (GET-VISUAL-PHYSICAL-PROPERTY OBJECT 'COLOR))
1146      (SETQ X (LOCATION-X (GET-VISUAL-OBJECT-PHYSICAL-LOCATION OBJECT)))
1147      (SETQ Y (LOCATION-Y (GET-VISUAL-OBJECT-PHYSICAL-LOCATION OBJECT)))
1148
1149      (CASE SHAPE
1150          (SQUARE (DRAW-VISUAL-OBJECT 'OPEN-SQUARE X Y SIZE))
1151          (CIRCLE (DRAW-VISUAL-OBJECT 'OPEN-CIRCLE X Y SIZE))
1152          (CROSS (DRAW-VISUAL-OBJECT 'CROSSHAIRS X Y SIZE))
1153          (OTHERWISE
1154              (COND
1155                  ((AND SHAPE COLOR)
1156                   (DRAW-VISUAL-OBJECT 'TEXT X Y SHAPE)
1157                   (DRAW-VISUAL-OBJECT 'TEXT (- X 1.5) (+ Y 1.) COLOR)
1158                  )
1159              (T (DRAW-VISUAL-OBJECT 'TEXT X Y TEXT))))
1160      )
1161 )
1162
1163 ))
1164
1165

```