```
 1  %donald
 2  viewport view
 3  #       The following definitions define a room ...
 4  #       ... and some objects inside the room.
 5  #
 6
 7  # The room is rectangle in shape.
 8
 9  int     width, length           # the dimension of room
10  point   NW, NE, SW, SE          # the 4 corners of room
11  line    N1, N2, S, E, W         # the 4 walls of room
12                                  # North wall composites of 2 walls & a
                                    door
13
14                                  ### Every room has a door ###
15  openshape door                  # declare the door
16
17  within door {
18          point   hinge, lock     # a door has a hinge, and a lock
19          line    door            # the door itself
20          int     width           # the size of the door
21          boolean open            # use it as a flag, telling whether
            ...
22                                  # ... the door has opened
23          door = [hinge, lock]
24
25          lock = hinge + (if open then {0, -width} else {width, 0})
26  }
27
28  within door {
29          hinge = ~/NW + {15, -10}        # set the coordinate of the
            hinge
30          open = true                     # the door has opened
31          width = 200                     # the size of the door
32  }
33
34  N1 = [NW, {door/hinge.1, NW.2}]
35  N2 = [{door/hinge.1+door/width, NW.2}, NE]
36  S = [SW, SE]
37  W = [NW, SW]
38  E = [NE, SE]
39
40  SW = {100, 100}
41  SE = SW + {width, 0}            # The other 3 corners are relative
42  NE = SW + {width, length}       # to South-West corner.
43  NW = SW + {0, length}           #
44
45  width, length = 800, 800        # 800*800 pts
46
47  ##########################################################################
    #####
48  openshape table                 # There is a table inside the room.
49
50  within table {
51          int     width, length           # the dimension of table
52          point   NW, NE, SW, SE          # the 4 corners of table
53          line    N, S, E, W              # the 4 sides of table
54
55          N = [NW, NE]
56          S = [SW, SE]
57          W = [NW, SW]
58          E = [NE, SE]
59
60          SE = SW + {width, 0}            # the other 3 corners are
61          NE = SW + {width, length}       # relative to SW corner
62          NW = SW + {0, length}           #
63
64          ##### table/lamp #####
65
66          openshape lamp                  # There is a lamp on the table
67          within lamp {
68                  point   centre          # center of the table
69                  int     size            # size of the lamp
```

```
 70                  int     half            # half of size
 71                  circle  base
 72
 73                  size = 50
 74                  half = size div 2
 75                  centre = (~/SW + ~/NE) div 2    # at the centre of
                     table
 76
 77                                          # L1-L8 forms an octagon.
 78                  line    L1, L2, L3, L4, L5, L6, L7, L8
 79
 80                  L1 = [centre + {size, -half}, centre + {size, half}]
 81                  L2 = [L1.2, centre + {half, size}]
 82                  L3 = [L2.2, centre + {-half, size}]
 83                  L4 = [L3.2, centre + {-size, half}]
 84                  L5 = [L4.2, centre + {-size, -half}]
 85                  L6 = [L5.2, centre + {-half, -size}]
 86                  L7 = [L6.2, centre + {half, -size}]
 87                  L8 = [L7.2, L1.1]
 88                  base = circle(centre, size * 1.25)
 89          }
 90  }
 91
 92  within table {
 93                  # let the size of table be 300*300 pts.
 94          width, length = 300, 300
 95
 96                  # place the SW of table at the center of the room.
 97          SW = (~/SW + ~/NE) div 2
 98  }
 99
100  ##########################################################################
     #####
101  point   plug
102  point   plug1, plug2            # there are 2 plugs.
103  plug1 = (S.1+S.2) div 2         # middle of South wall
104  plug2 = (E.1+E.2) div 2         # middle of East wall
105
106  line    cable                   # cable connects the table/lamp and
     the plug
107  int     cablelength
108  cable = [plug, table/lamp/centre]
109  plug = plug1                    # let cable connects to plug1
110  cablelength = 600
111  #
112  # Now set the line style of cable be dotted line.
113  #   Because many Donald commands hasn't implemented,
114  #       I just directly access Eden using the non-standard `?'
     command.
115  ? A_cable = "linestyle=dashed,dash=13";
116
117
118  ##########################################################################
     #####
119
120  openshape desk                  # there is a desk in the room
121
122  within desk {
123          int     width, length           # the size of the desk
124          point   NW, NE, SW, SE          # the 4 corners of the desk
125          line    N, S, E, W              # the 4 edges of the desk
126
127          N = [NW, NE]
128          S = [SW, SE]
129          W = [NW, SW]
130          E = [NE, SE]
131
132          width, length = 250, 350
133
134                  # initially the desk is placed at (100,100) of the
                     room.
135          SW = ~/SW + {15, 15}
136          SE = SW + {width, 0}            # the other 3 corners are ...
```

```
137          NW = SW + {0, length}           # ... relative to SW corner
138          NE = NW + {width, 0}            #
139
140          ##### desk/drawer #####
141
142          openshape drawer        # the desk has a drawer
143
144          within drawer {
145                  int     k       # k --  a parameter describes
146                                  #       the condition of drawer
147                                  #               1 --> closed
148                                  #               larger k --> more open
149
150                  int     width, length   # the size of drawer
151                  point   NW, NE, SW, SE  # the 4 corners of the drawer
152                  line    N, S, W, E      # the 4 edges of the drawer
153
154                          # the size of the drawer is always a ratio to
                             ...
155                          # ... the desk.
156                  width = ~/length div 3
157                  length = ~/width - ~/width div k
158                  k = 2                   # initially, the draw is half
                     open
159
160                  NW = ~/NE               # NW is always at desk's NE
161                  SW = NW - {0, width}    # the other 3 corners are ...
162                  SE = SW + {length, 0}   # ... relative to NW
163                  NE = NW + {length, 0}   #
164
165                  N = [NW, NE]
166                  S = [SW, SE]
167                  W = [NW, SW]
168                  E = [NE, SE]
169          }
170  }
171
172  boolean doorHitTable, cableIsShort
173  doorHitTable = includes(circle(door/hinge, door/width),table/NW)
174  cableIsShort = dist(cable.1, cable.2) > cablelength
175
176  %scout
177
178  display scr, basicScreen;
179  window don1, don2;
180  window monDoor, monCable;
181  point monDoorPos, monCablePos;
182  string monDoorStr, monCableStr;
183  integer _doorHitTable, _cableIsShort, _door_open;
184  integer _plug, _plug1;
185  point p1, q1, p2, q2;
186  integer zoomSize;
187  point zoomPos;
188  point tblMenuRef, miscMenuRef, zoomMenuRef;
189  point plugButtonPos, doorButtonPos;
190  string plugMenu, doorMenu;
191  window plugButton, doorButton;
192  point tblMenuHeaderPos, tblUpPos, tblDownPos, tblLeftPos, tblRightPos;
193  string tblMenuHeader, tblUpMenu, tblDownMenu, tblLeftMenu,
     tblRightMenu;
194  window tblHeader, tblUp, tblDown, tblLeft, tblRight;
195  point zoomMenuHeaderPos, zoomUpPos, zoomDownPos, zoomLeftPos,
     zoomRightPos;
196  string zoomMenuHeader, zoomUpMenu, zoomDownMenu, zoomLeftMenu,
     zoomRightMenu;
197  window zoomHeader, zoomUp, zoomDown, zoomLeft, zoomRight;
198
199  p1 = {25, 100};
200  q1 = {225, 300};
201  don1 = {
202          box:    [p1, q1],
203          pict:   "view",
204          type:   DONALD,
```

```
205          border: 1
206          sensitive: ON
207  };
208  zoomPos = {500, 500};
209  zoomSize = 500;
210  p2 = {275, 100};
211  q2 = {475, 300};
212  don2 = {
213          box:    [p2, q2],
214          pict:   "view",
215          type:   DONALD,
216          xmin:   zoomPos.1 - zoomSize/2,
217          ymin:   zoomPos.2 - zoomSize/2,
218          xmax:   zoomPos.1 + zoomSize/2,
219          ymax:   zoomPos.2 + zoomSize/2,
220          border: 1
221          sensitive: ON
222  };
223
224  monDoor = {
225          frame:  ([monDoorPos, 1, strlen(monDoorStr)]),
226          string: monDoorStr
227  };
228  monDoorStr = if _doorHitTable then "Table obstructs door" else ""
     endif;
229  monDoorPos = {25, 50};
230
231  monCable = {
232          frame:  ([monCablePos, 1, strlen(monCableStr)]),
233          string: monCableStr
234  };
235  monCableStr = if _cableIsShort then "Cable is not long enough" else ""
     endif;
236  monCablePos = {25, 70};
237
238  tblMenuRef = {100, 400};
239  miscMenuRef = {250, 400};
240  zoomMenuRef = {400, 400};
241
242  plugButtonPos = miscMenuRef - {strlen(plugMenu).c / 2, 1.r};
243  plugButton = {
244          frame:  ([plugButtonPos, 1, strlen(plugMenu)]),
245          string: plugMenu,
246          border: 1
247          sensitive: ON
248  };
249  plugMenu = if _plug == _plug1 then "9:Use Plug 2" else "9:Use Plug 1"
     endif;
250
251  doorButtonPos = miscMenuRef + {-strlen(doorMenu).c / 2, 1.r};
252  doorButton = {
253          frame:  ([doorButtonPos, 1, strlen(doorMenu)]),
254          string: doorMenu,
255          border: 1
256          sensitive: ON
257  };
258  doorMenu = if _door_open then "10:Close Door" else "10:Open Door"
     endif;
259
260  tblMenuHeader = "Table Position";
261  tblMenuHeaderPos = tblMenuRef - {(strlen(tblMenuHeader)/2).c, 4.r};
262  tblHeader = {
263          frame:  ([tblMenuHeaderPos, 1, strlen(tblMenuHeader)]),
264          string: tblMenuHeader
265  };
266
267  tblUpPos = tblMenuRef - {(strlen(tblUpMenu)/2).c, 2.r};
268  tblUp = {
269          frame:  ([tblUpPos, 1, strlen(tblUpMenu)]),
270          string: tblUpMenu,
271          border: 1
272          sensitive: ON
273  };
```

```
274  tblUpMenu = "1:Up";
275
276  tblDownPos = tblMenuRef + {-(strlen(tblDownMenu)/2).c, 2.r};
277  tblDown = {
278          frame:  ([tblDownPos, 1, strlen(tblDownMenu)]),
279          string: tblDownMenu,
280          border: 1
281          sensitive: ON
282  };
283  tblDownMenu = "2:Down";
284
285  tblLeftPos = tblMenuRef - {(strlen(tblLeftMenu) + 1).c, 0};
286  tblLeft = {
287          frame:  ([tblLeftPos, 1, strlen(tblLeftMenu)]),
288          string: tblLeftMenu,
289          border: 1
290          sensitive: ON
291  };
292  tblLeftMenu = "3:Left";
293
294  tblRightPos = tblMenuRef + {1.c, 0};
295  tblRight = {
296          frame:  ([tblRightPos, 1, strlen(tblRightMenu)]),
297          string: tblRightMenu,
298          border: 1
299          sensitive: ON
300  };
301  tblRightMenu = "4:Right";
302
303  zoomMenuHeader = "Zoom Position";
304  zoomMenuHeaderPos = zoomMenuRef - {(strlen(zoomMenuHeader)/2).c, 4.r};
305  zoomHeader = {
306          frame:  ([zoomMenuHeaderPos, 1, strlen(zoomMenuHeader)]),
307          string: zoomMenuHeader
308  };
309
310  zoomUpPos = zoomMenuRef - {(strlen(zoomUpMenu)/2).c, 2.r};
311  zoomUp = {
312          frame:  ([zoomUpPos, 1, strlen(zoomUpMenu)]),
313          string: zoomUpMenu,
314          border: 1
315          sensitive: ON
316  };
317  zoomUpMenu = "5:Up";
318
319  zoomDownPos = zoomMenuRef + {-(strlen(zoomDownMenu)/2).c, 2.r};
320  zoomDown = {
321          frame:  ([zoomDownPos, 1, strlen(zoomDownMenu)]),
322          string: zoomDownMenu,
323          border: 1
324          sensitive: ON
325  };
326  zoomDownMenu = "6:Down";
327
328  zoomLeftPos = zoomMenuRef - {(strlen(zoomLeftMenu) + 1).c, 0};
329  zoomLeft = {
330          frame:  ([zoomLeftPos, 1, strlen(zoomLeftMenu)]),
331          string: zoomLeftMenu,
332          border: 1
333          sensitive: ON
334  };
335  zoomLeftMenu = "7:Left";
336
337  zoomRightPos = zoomMenuRef + {1.c, 0};
338  zoomRight = {
339          frame:  ([zoomRightPos, 1, strlen(zoomRightMenu)]),
340          string: zoomRightMenu,
341          border: 1
342          sensitive: ON
343  };
344  zoomRightMenu = "8:Right";
345
346  basicScreen = < tblHeader / tblUp / tblDown / tblLeft / tblRight /
```

```
347             zoomHeader / zoomUp / zoomDown / zoomLeft / zoomRight /
348             plugButton / doorButton /
349             don1 / don2 >;
350
351   scr = if _doorHitTable then
352           append(basicScreen, 1, monDoor)
353         else
354           basicScreen
355         endif;
356
357   #screen = < monDoor / monCable /
358   #          tblHeader / tblUp / tblDown / tblLeft / tblRight /
359   #          zoomHeader / zoomUp / zoomDown / zoomLeft / zoomRight /
360   #          plugButton / doorButton /
361   #          don1 / don2 >;
362
363   screen = if _cableIsShort then
364                   append(scr, 1, monCable)
365            else
366                   scr
367            endif;
368
369   %eden
370   proc handle_user_input : input
371   {
372           switch (input) {
373           case 1: _table_SW = vector_add(_table_SW, cart(0, 100));
                    break;
374           case 2: _table_SW = vector_sub(_table_SW, cart(0, 100));
                    break;
375           case 3: _table_SW = vector_sub(_table_SW, cart(100, 0));
                    break;
376           case 4: _table_SW = vector_add(_table_SW, cart(100, 0));
                    break;
377           case 5: zoomPos = pt_add(zoomPos, [0, 100]);
                    break;
378           case 6: zoomPos = pt_subtract(zoomPos, [0, 100]);
                    break;
379           case 7: zoomPos = pt_subtract(zoomPos, [100, 0]);
                    break;
380           case 8: zoomPos = pt_add(zoomPos, [100, 0]);
                    break;
381           case 9: if (_plug == _plug1) {
382                           _plug is _plug2;
383                   } else {
384                           _plug is _plug1;
385                   }
386                   break;
387           case 10: _door_open = !_door_open;
                    break;
388           }
389   }
390
391   proc plugButton_to_input : plugButton_mouse_1 {
392           if (plugButton_mouse_1[2] == 4) input = 9;
393   }
394
395   proc doorButton_to_input : doorButton_mouse_1 {
396           if (doorButton_mouse_1[2] == 4) input = 10;
397   }
398
399   proc tblUp_to_input : tblUp_mouse_1 {
400           if (tblUp_mouse_1[2] == 4) input = 1;
401   }
402
403   proc tblDown_to_input : tblDown_mouse_1 {
404           if (tblDown_mouse_1[2] == 4) input = 2;
405   }
406
407   proc tblLeft_to_input : tblLeft_mouse_1 {
408           if (tblLeft_mouse_1[2] == 4) input = 3;
409   }
410
```

```
411  proc tblRight_to_input : tblRight_mouse_1 {
412          if (tblRight_mouse_1[2] == 4) input = 4;
413  }
414
415  proc zoomUp_to_input : zoomUp_mouse_1 {
416          if (zoomUp_mouse_1[2] == 4) input = 5;
417  }
418
419  proc zoomDown_to_input : zoomDown_mouse_1 {
420          if (zoomDown_mouse_1[2] == 4) input = 6;
421  }
422
423  proc zoomLeft_to_input : zoomLeft_mouse_1 {
424          if (zoomLeft_mouse_1[2] == 4) input = 7;
425  }
426
427  proc zoomRight_to_input : zoomRight_mouse_1 {
428          if (zoomRight_mouse_1[2] == 4) input = 8;
429  }
430
431  proc don1_to_tableSW : don1_mouse {
432      auto mx, my;
433      mx = don1_mouse[4];
434      my = don1_mouse[5];
435      if (don1_mouse[2] == 4) {
436          if (_table_SW[2] < mx && mx < _table_NE[2] &&
437              _table_SW[3] < my && my < _table_NE[3]) move_table = 1;
438          old_table_SW = _table_SW;
439          old_mouse_pos = [mx, my];
440      }
441      if (don1_mouse[2] == 5) {
442          if (move_table == 1) {
443              _table_SW = cart(old_table_SW[2] + mx - old_mouse_pos[1],
444                               old_table_SW[3] + my - old_mouse_pos[2]);
445              move_table = 0;
446          }
447      }
448  }
449
450  proc don2_to_tableSW : don2_mouse {
451      auto mx, my, xmin, xmax, ymin, ymax, m, c, cpi;
452      if (don2_mouse[2] == 4) {
453          mx = don2_mouse[4];
454          my = don2_mouse[5];
455          xmin = dotint(don2, 6);
456          ymin = dotint(don2, 7);
457          xmax = dotint(don2, 8);
458          ymax = dotint(don2, 9);
459          m = (ymax - ymin) / (xmax - xmin);
460          c = ymin - m * xmin;
461          cpi = ymax + m * xmin;
462          if ((my - m * mx) > c)
463                  input = ((my + m * mx) > cpi) ? 5 : 7;
464          else
465                  input = ((my + m * mx) > cpi) ? 8 : 6;
466      }
467  }
468
```