RECOGNITION MEMORY EXPERIMENT FRAMEWORK

DESIGNERS:
M. RABE MMRABE@UVIC.CA
DR. S. LINDSAY SLINDSAY@UVIC.CA

DEVELOPER:
A. RICHARDSON
RICHARDSON.ASHLIN@GMAIL.COM

INSTITUTION:
UNIVERSITY OF VICTORIA

Contents

Date: May 12, 2017.

Overview

An online framework for parametric generation of Recognition Memory experiments to support researchers at the University of Victoria. The software is web based, self contained yet comprehensive, and reasonably flexible.

0.1. **Requirements.**

Server-side.

- Host:
  - An ordinary web server with Python/CGI enabled, is required.
  - Note: the system was tested with server: Apache/2.2.23 (Unix).

Client-side.

- For experiment participants:
  - A modern web browser (Firefox, Google Chrome, or Safari) on a desktop computer is required.
  - Note: the system was tested with Chrome v. 57.

- For administrators:
  - An FTP program is required for uploading experiment scripts (and downloading response data).
  - A text editor is required to edit experiment script files.
  - Limited technical knowledge about JavaScript is required to edit or modify experiments.

1. The System

The system, which may be downloaded from

https://github.com/ashlinrichardson/m3m0ry/archive/master.zip

has the following directory structure:

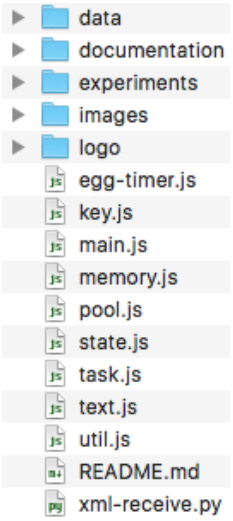

FIGURE 1.1.

where this document lives in the documentation/ folder. Additionally,

- **data**/ will contain CSV data files representing the user experience.
  - If all goes well, a data file should automagically appear in the **data**/ folder, each time a survey/experiment is completed.

  - Upon completion of a survey/experiment, the client-side JavaScript code submits (via util.js::xml_send()) a CSV data file to the web server, which receives the data using CGI/Python (via xml-receive.py).
  - The CSV file is saved with a name reflecting the date/time when the file was recorded, and a randomly-generated string to prevent "collisions".

- **images**/ contains image data used in experiments. To change the image data used in experiments, the administrator should:
  - upload new image data into the **images**/ folder, and
  - modify (an) experiment script(s) to reflect the filenames corresponding to the new image files added.
    - This is slightly technical, unless the image data obeys the usual numbered file-name convention.
- **experiments**/
  - contains a number of sub-folders, one for each of the included examples:
    - delay/
    - feedback/
    - instructions/
    - study-phase/
    - test-phase/
    - my-experiment/
  - Each subfolder contains a file **memory.html**, which always has the contents:

6 lines (5 sloc) | 70 Bytes

```
1  <html>
2  <body>
3  <script src="../../memory.js"></script>
4  </body>
5  </html>
```

FIGURE 1.2.

  - Then, supposing the project is uploaded to the main HTTP directory of a web server with URL **http://my-web-server.com**, the survey in the folder experiments/my-experiment/ represented by **experiments/my-experiment/my-experiment.js** will be accessed by navigating to the following address, in a web browser:
    - **http://my-web-server.com/experiments/my-experiment/memory.html**
  - **To create your own experiment,** we recommend editing the file **my-experiment.js** within the **my-experiment**/ folder
    - **To deploy your experiment on the web,** don't forget to upload your revised my-experiment.js to the server.

2. The Examples

Need to add

x.set_expiry(delay_time)

x.key_expiry = false

2.1. **experiments/instructions.**

```
/* recognition memory experiment set-up */
var my_experiment = function(){

  /* instruction slide */
  instructions('welcome to the recognition memory experiment framework (press any key to continue)')

  /* instruction slide */
  instructions('here is what happens when you put in a lot of text— if you put in lots of text, it might g

  /* instruction slide */
  instructions('this is an instructions slide (press any key to continue)')

  /* instruction slide */
  instructions('this is another instructions slide (press any key to continue)')

  /* instruction slide -- fixed duration */
  var x = instructions('this instructions slide will display for 5 seconds: if you press a key, it will do
  x.set_expiry(5000)
  x.key_expiry = false

  /* instruction slide -- fixed duration or user intervention */
  var y = instructions('this instructions slide will display for up to 5 seconds: if you press a key, the t
  y.set_expiry(5000)
  y.key_expiry = true

  /* instruction slide */
  instructions('this is a normal instructions slide (press any key to continue)')

}
```

2.2. **experiments/delay.**

```
/* recognition memory experiment set-up */
var my_experiment = function(){

  /* instruction slide */
  instructions('first delay phase (5 seconds): please press any key to start')

  /* set up delay task: 5 seconds */
  delay_task('please type names of as many countries as you can think of in 5 seconds, separated by spaces

  /* instruction slide */
  instructions('second delay phase (30 seconds): please press any key to start')

  /* set up delay task: 30 seconds */
  delay_task('please type names of as many countries as you can think of in 30 seconds, separated by spaces

  /* instruction slide */
  instructions('all done.. thank you')
}
```

2.3. **experiments/feedback.**

```
/* recognition memory experiment set-up */
var my_experiment = function(){

  /* instructions */
  instructions('feedback coming up... please press any key...')

  /* feedback "task" */
  feedback('please enter your affinity with the last stimulus on a scale of 1-5', [49, 50, 51, 52, 53])

  /* instructions */
```

```
  instructions('thank you... more feedback coming up... please press any key...')

  /* more feedback "task" */
  feedback('please enter your affinity with the last stimulus on a scale of 0-9', [49, 50, 51, 52, 53, 54,

  /* instructions */
  instructions('thank you... multiple choice style feedback coming up... please press any key...')

  /* feedback "task" */
  feedback('skill testing question: 10*10 is: a) 100 b) 200 c) 1000 d) 10000', [65, 66, 67, 68])

  /* instructions */
  instructions('thank you.. please press any key to finish')
}
```

2.4. **experiments/study-phase.**

```
/* recognition memory experiment set-up */
var my_experiment = function(){

  /* instructions */
  instructions('study phase coming next:')
  instructions('please remember each word/image and press any key')

  /* set up a stimulus pool */
  var p = stimulus_pool()

  /* add images to stimulus pool */
  for(var i = 0; i < 10; i++){
    p.add(ctx.imgs[i])
  }

  /* add words to stimulus pool */
  p.add('floccinaucinihilipilification')
  p.add('supercalifragilisticexpialidocious')
  p.add('umdiddlediddlediddleumdiddlei')

  /* select portion of items from stimulus pool */
  p.select(3, 3)

  /* set up 'study phase': show selected portions of pool */
  study_phase(p, 111)
}
```

2.5. **experiments/test-phase.**

```
/* recognition memory experiment set-up */
var my_experiment = function(){

  /* set up some instruction slides */
  instructions('study phase: please remember images and press any key')

  /* set up a stimulus pool */
  var p = stimulus_pool()

  /* add images to stimulus pool */
  for(var i = 0; i < 10; i++){
    p.add(ctx.imgs[i])
  }

  /* add words to stimulus pool */
  p.add('floccinaucinihilipilification')
  p.add('supercalifragilisticexpialidocious')
  p.add('umdiddlediddlediddleumdiddlei')
```

```
19
20   /* selection from stimulus pool (parameters are N, M) */
21   p.select(3, 3)
22
23   /* set up 'study phase': show selected portions of pool */
24   study_phase(p, 111)
25
26   /* some instructions before 'test phase' */
27   instructions('test phase coming up')
28   instructions('when you see an image/word, please press m or n')
29   instructions('please press m if you saw an image/word before')
30   instructions('please press n if you did not see the image/word before')
31
32   /* set up 'test phase' (user input recorded for whole randomized pool) */
33   test_phase(p, 333)
34 }
```

3. Sample Response Data

4. Source Code: Client Side

4.1. egg-timer.js.

```
1  /* via developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/clearTimeout */
2  var egg_timer = {
3
4    /* callback */
5    setup: function(t_ms){
6
7      /* assert parameter is a number */
8      if(typeof this.timeoutID === "number"){
9        this.cancel()
10     }
11
12     /* what to do when the timer expires */
13     this.timeoutID = window.setTimeout(function(){
14         var now = ctx.get_state()
15         var id = now.id
16         /* console.log('ding from now(): id', id) */
17         now.ding = true
18         if(now.key_expiry ==false){
19           now.expire()
20         }
21     }.bind(this), t_ms)
22   }, cancel: function() {
23     window.clearTimeout(this.timeoutID)
24     this.timeoutID = undefined
25   }
26 }
```

4.2. key.js.

```
1  /* convert form unicode to familiar symbol */
2  function unicode_from_key_event(e){
3    return e.charCode? e.charCode : e.keyCode
4  }
5
6  /* keyboard status array (unicode format) */
7  var key_unicode = {}
8
9  /* keyboard event handler function */
10 function keyboard_module(){
11
```

```
12  /* set up key-down event handler function */
13  document.onkeydown = function(e){
14    var unicode = unicode_from_key_event(e), key =  String.fromCharCode(unicode)
15    key_unicode[unicode] = true
16
17    console.log("unicode", unicode)
18
19    /* ignore caps-lock key */
20    if(unicode == 20){
21
22      /* enable this line to debug key codes: console.log("unicode", unicode) */
23      return
24    }
25
26    /* when are we? */
27    var now = ctx.get_state()
28
29    /* record key press, if admissible */
30    var admissible_keys = now.get_admissible_keys()
31    if(admissible_keys.includes(unicode)){
32      now.record_key_stroke(unicode)
33    }
34
35    /* by default, transition from a slide upon key-press */
36    var go = true
37    if(now.type=='delay'){
38      if(now.txt ==null){
39        now.txt =''
40      }
41      if(unicode ==8 ){
42        var len = now.txt.length
43        if(now.txt[len-1] != ' '){
44          now.txt = now.txt.substring(0, len - 1)
45        }
46      }else if(unicode == 0){
47      }else{
48        now.txt += key.toLowerCase()
49      }
50      update()
51    }
52
53    /* check if this state "requires" keyboard input */
54    if(now.require_key() == true){
55      if(admissible_keys.includes(unicode)){
56        if(!(now.deja == undefined)){
57          ctx.questions_total += 1
58          if((now.deja == true && unicode == 77)||(now.deja == false && unicode == 78)){
59            ctx.questions_correct += 1
60          }
61        }
62      }else{
63
64        /* block if a key was required but the one entered was not admissible */
65        go = false
66      }
67    }
68    if(now.ding==false && now.hold==true){
69      go = false
70    }
71
72    /* t <-- t + 1 */
73    if(now && now.key_expiry && go){
74        ctx.clear_tmr()
75        now.expire()
76    }
```

```
77   }
78   return key_unicode
79 }
```

4.3. **main.js.**

```
1  abs_path = '../../'
2  var history = [], canvas = document.getElementsByTagName("canvas")[0], ctx = canvas.getContext("2d")
3
4  /* background color */
5  document.bgColor = "#FFFFFF"
6
7  /* shape parameter */
8  ctx.pad = 20
9
10 /* font size */
11 ctx.font_size = 30
12
13 /* canvas dimensions manipulation */
14 var less = function(x){
15   return x - ctx.pad
16 }
17
18 ctx.w = function(){
19   return less(window.innerWidth)
20 }
21
22 ctx.h = function(){
23   return less(window.innerHeight)
24 }
25
26 /* canvas resize */
27 function resize(){
28   canvas.width = ctx.w()
29   canvas.height = ctx.h()
30 }
31
32 /* load corporate logo */
33 ctx.symbol = load_img(abs_path + "logo/uvic_gray.png")
34
35 /* algo to draw scaled corporate logo */
36 ctx.draw_symbol = function(){
37   var s_f = 5, pad = this.pad, s = this.symbol
38   var ww = window.innerWidth, wh = window.innerHeight
39   var w = ww - pad, h = wh - pad, w_s = s.width, h_s = s.height
40   var wf = (ww - pad) / (s_f * w_s), lwf = w_s * wf, lhf = h_s * wf
41   this.drawImage(s, w - lwf, h - lhf, lwf, lhf)
42 }
43
44 /* access current "state" (a state represents a particular "trial" in an experiment) */
45 ctx.set_state = function(s){
46   last_state = null;
47   if(ctx.current_state != null){
48     last_state = ctx.current_state
49   }
50   ctx.current_state = s
51
52   /* should not happen.. */
53   if(s != null){
54     s.daddy = last_state
55   }
56   return(s)
57 }
58
```

```
/* access present "state" */
ctx.get_state = function(){
  var s = ctx.current_state
  var st = ''
  try{
    st = s.txt
  }catch(e){
    st = ''
  }
  return s
}

/* trigger update/plotting from window resize event */
window.onresize = function(event){
  update()
}

/* update the canvas (present the current "trial") */
function update(){
  resize()
  var now = ctx.get_state()
  if(now != null)
    now.show(ctx)
}

/* "in" hook: plot the current trial */
window.onload = function(){
  update()
}

/* set up timer to coordinate transitions between trials */
ctx.egg_timer = egg_timer

ctx.clear_tmr = function(){
  ctx.egg_timer.cancel()
}

ctx.init_tmr = function(t_ms){
  ctx.egg_timer.setup(t_ms)
}

/* initialize reference to first and most-recently-initialized trials */
ctx.last_new_state = null
ctx.first_new_state = null

/* count number of questions answered correctly (this is redundant) */
ctx.questions_correct = 0
ctx.questions_total = 0

/* this function sets up the experiment (according to the user function my_experiment)
and we trigger this function after all the images have loaded. */
function run_after_loading_images(){

  /* set up an experiment according to user specs/code */
  my_experiment(ctx)

  /* in this part, we should record only the images that we actually need */

  instructions('thank you')

  ctx.last_state = ctx.last_new_state
  ctx.first_state = ctx.first_new_state

  /* start at the very beginning, it's a very good place to start.. */
  ctx.set_state(ctx.first_state)
```

```
124
125   /* respond to keyboard events */
126   key_unicode = keyboard_module()
127
128   /* start "stopwatch" */
129   ctx.t0 = window.performance.now()
130
131   /* go */
132   ctx.get_state().start()
133 }
134
135 /* load some image files: need to change if the image database changes */
136 var n_imgs = 200
137 var n_imgs_loaded = 0
138
139 /* load image data */
140 function load_img(fn){
141   var img = new Image()
142   img.onload = function(){
143     /* console.log('loaded image: ', fn) */
144     n_imgs_loaded += 1
145     if(n_imgs_loaded == n_imgs){
146
147       /* proceed to init the experiment, after all images loaded.. */
148       run_after_loading_images()
149     }
150   }
151   /* load the image */
152   img.src = fn
153   return img
154 }
155
156 /* load all of the image data */
157 ctx.load_imgs = function(n_imgs){
158
159   /* ideally would only load the ones used */
160   var imgs = new Array()
161   for(var i=1; i <= n_imgs; i++){
162     var img_fn = abs_path + 'images/' + i + '.jpg'
163     var my_img = load_img(img_fn)
164     my_img.fn = 'images/' + i + '.jpg'
165     imgs.push(my_img)
166   }
167   ctx.imgs = imgs
168   return ctx.imgs
169 }
170
171 /* keep track of the "task-index" as the experiment is intialized */
172 var next_task_id = 0
173
174 /* this line "makes everything go" */
175 var my_images = ctx.load_imgs(n_imgs)
```

4.4. **memory.js.**

```
1 /* sleep function */
2 function sleep(ms){
3   return new Promise(resolve => setTimeout(resolve, ms))
4 }
5
6 /* cr34t3 a c4nv4s wh3r3 th3 m4g1c h4pp3ns */
7 var canvas = document.createElement('canvas')
8 document.body.appendChild(canvas)
9 var js_added = 0
```

```
deps = []

/*  j4v4scr1pt 4n4l0g 0f 1nclud3 st4t3m3nt */
function add_js(fn){
  var body = document.getElementsByTagName('body')[0], s = document.createElement('script')
  s.async = false
  s.src = fn + '.js'
  var callback = function(){
    js_added += 1
    if(js_added < deps.length){
      add_js(deps[js_added])
    }
  }

  /* wait until script is loaded before proceeding.. */
  s.onload = callback
  var len = body.childNodes.length
  body.appendChild(s)
}

/* c411 411 th3 ch1ldr3n */
dependencies = ['text', 'key', 'util', 'task', 'pool', 'state', 'egg-timer']
for(var d in dependencies){
  deps.push('../../' + dependencies[d])
}
deps.push('my-experiment')
deps.push('../../main')
add_js(deps[0], '')
```

4.5. **pool.js.**

```
/* stimulus pool - object that has words or images added to it. Selections drawn randomly for "study phase"
var next_pool_id = 0
function pool(){
  this.is_pool = true
  this.pool_id = next_pool_id
  next_pool_id += 1
  console.log('pool, id=', this.pool_id)
  this.ctx = ctx
  this.stimuli = new Array()

  /* add a stimulus to the pool */
  this.add = function(stim){
    this.stimuli.push(stim)
    return stim
  }

  /* set number of samples for study phase */
  this.set_n = function(n){
    this.n = n
  }

  /* set number of additional samples to be included for test phase */
  this.set_m = function(m){

    /* subsequently to drawing "n" items from the pool (without replacement), an additional "m" samples are
    this.m = m
  }

  /* get */
  this.get_n = function(){
    return this.n
  }

```

```
34  /* get */
35  this.get_m = function(){
36    return this.m
37  }
38
39
40  /* remove any "blank" elements (an operation needed due to an apparent curiosity of the language) that ap
41  this.remove_blanks = function(){
42    this.stimuli = this.stimuli.filter(function(){return true})
43  }
44
45  /* pseudorandom selection of size "n" */
46  this.draw_n = function(){
47    console.log('\tpool, id=', this.pool_id)
48
49    if(this.selection_n){
50      console.log('error: n-selection already made from this pool.')
51      return null
52    }
53    var n = parseInt(get_n())
54    if(n > this.stimuli.length){
55      console.log('error: n > this.stimuli.length')
56      return null
57    }
58    this.selection_n = new Array()
59    var rem = this.stimuli.length
60    for(var i = 0; i < n; i++){
61      var qx = rand() * parseFloat(rem), idx = parseInt(qx)
62      rem -= 1
63      this.selection_n.push(this.stimuli[idx])
64      delete this.stimuli[idx]
65      this.remove_blanks()
66    }
67  }
68
69  /* pseudorandom selection of size "m" */
70  this.draw_m = function(){
71    console.log('\tpool, id=', this.pool_id)
72
73    if(this.selection_m){
74      console.log('error: m-selection already made from this pool.')
75      return null
76    }
77    var m = parseInt(get_m())
78    if(m > this.stimuli.length){
79      console.log('error: m > this.stimuli.length')
80      return null
81    }
82    this.selection_m = new Array()
83    var rem = this.stimuli.length
84    for(var i = 0; i < m; i++){
85      var qx = rand() * parseFloat(rem), idx = parseInt(qx)
86      rem -= 1
87      this.selection_m.push(this.stimuli[idx])
88      delete this.stimuli[idx]
89      this.remove_blanks()
90    }
91  }
92
93  /* for initializing a test phase: mix "N"-selection and "M"-selection together */
94  this.reshuffle = function(){
95    var to_shuffle = [], i = 0
96
97    /* add the "N"-selection */
98    for(i = 0; i < this.selection_n.length; i++){
```

```
99        var dat_i = new Array()
100       dat_i.push(this.selection_n[i])
101       dat_i.push(true)
102       to_shuffle.push(dat_i)
103     }
104
105     /* add the "M"-selection */
106     for(i = 0; i < this.selection_m.length; i++){
107       var dat_i = new Array()
108       dat_i.push(this.selection_m[i])
109       dat_i.push(false)
110       to_shuffle.push(dat_i)
111     }
112
113     /* "shuffle"-- randomize the ordering of the combined array */
114     var shuffled = new Array(), deja_vu = new Array(), rem = to_shuffle.length
115     while(rem >0){
116       rem -= 1
117       var idx = parseInt(rand() * parseFloat(rem)), dat_i = to_shuffle[idx]
118       shuffled.push(dat_i[0])
119       deja_vu.push(dat_i[1])
120       delete to_shuffle[idx]
121       to_shuffle = to_shuffle.filter(function(){return true})
122     }
123     var ret = [shuffled, deja_vu]
124     return ret
125   }
126
127   this.draw = function(){
128     console.log('draw_n()')
129     this.draw_n()
130     console.log('draw_m()')
131     this.draw_m()
132     console.log('reshuffle()')
133     this.reshuffle()
134   }
135
136   /* set N, M parameters and make a selection */
137   this.select = function(n,m){
138     console.log('select(n,m)')
139     this.set_n(n)
140     this.set_m(m)
141     this.draw()
142   }
143
144   /* end of "pool::pool()" */
145   return this
146 }
147
148 function stimulus_pool(){
149   return new pool()
150 }
```

4.6. **state.js.**

```
1 /* global counter for states/ AKA frames/ AKA slides */
2 var state_id = -1
3
4 function get_id(){
5   state_id += 1;
6   return state_id;
7 }
8
9 /* reference to 2d canvas graphics context */
```

```
10 function get_ctx(){
11   return document.getElementsByTagName("canvas")[0].getContext("2d");
12 }
13
14 /* state: generic object representing trial (like a card in "hypercard") */
15 function state(expiry_ms  =     0,  /* max. presentation time (mS) */
16               key_expiry =  true,  /* expiry by key-press (true <--> on) */
17               intvl_ms   =     0,  /* interval btwn stimuli.. (ISI) 'blank slide' */
18               img_idx    =    -1,  /* image data (if any) */
19               txt    =    null,  /* text data (if any) */
20               successor = null){
21   this.action = null
22   this.ding = false
23   var ctx = get_ctx()
24   this.hold = false
25
26   this.hold_on = function(){
27     this.hold = true
28   }
29   this.id = get_id()
30   this.key_required = false
31
32   /* array to store admissible key-codes for data entry or transition to next "slide" */
33   this.admissible_keys = [77,78]
34
35   this.get_admissible_keys = function(){
36     return this.admissible_keys
37   }
38
39   this.clear_admissible_keys = function(){
40     this.admissible_keys = new Array()
41   }
42
43   this.add_admissible_key = function(k){
44     this.admissible_keys.push(k)
45   }
46
47   /* this array will record the keystroke data received while residing in this state */
48   this.key_strokes = new Array()
49
50   this.record_key_stroke = function(k){
51     this.key_strokes.push(k)
52   }
53
54   this.set_pool_id = function(pid){
55     this.pool_id = pid
56   }
57   this.get_pool_id = function(){
58     if(this.pool_id)
59       return this.pool_id
60     else
61       return ""
62   }
63
64   /* keep a reference to this state, if it's the first one ever.. */
65   if(ctx.first_new_state == null){
66     ctx.first_new_state = this
67   }
68
69   /* only applies if there's a "next" trial, if this is a trial */
70   this.intvl_ms = intvl_ms
71
72   /* numeric */
73   this.expiry_ms = expiry_ms
74
```

```
75  /* boolean */
76  this.key_expiry = key_expiry
77
78  /* global image index (images added as member of ctx) */
79  this.img_idx = img_idx
80  this.successor = null
81
82  this.require_key = function(){
83    return this.key_required
84  }
85  this.predecessor = ctx.last_new_state;
86  var id = this.predecessor == null ? -1 : this.predecessor.id
87  ctx.last_new_state = this
88  if(this.predecessor != null){
89    this.predecessor.set_successor(this)
90  }
91
92  /* where are we going? */
93  this.set_successor = function(s){
94    this.successor = s
95  }
96
97  /* plot text or images */
98  this.show = function(){
99    if(this.action){
100     this.action(this)
101   }
102   var ctx = get_ctx()
103   ctx.clearRect(0, 0, ctx.w(), ctx.h())
104
105   /* bottom text */
106   if(this.txt2 && (!this.wrd_stim)){
107     //wrap_text(this.txt2, ctx, ctx.h() - (2 * ctx.font_size+20));
108   }
109   if(this.txt2){
110     wrap_text(this.txt2, ctx, ctx.h() - (2 * ctx.font_size + 20))
111   }
112
113   /* upper text */
114   if(this.txt){
115     wrap_text(this.txt, ctx, 0)
116   }
117
118   /* img or middle text (if word stim) */
119   if(this.img_stim){
120     x = this.img_stim
121     draw_img(x, ctx)
122   }
123
124   /* might need the wrap_text back on for long strings.. */
125   if(this.wrd_stim!=null){
126     // wrap_text(this.wrd_stim, ctx, ctx.h()/2);
127
128     /* no wrap */
129     centre_text(this.wrd_stim)
130   }
131
132   /* logo of no image/ lower text present */
133   if(!this.txt2){
134     ctx.draw_symbol()
135   }
136 }
137
138 /* state expires by timer or key press */
139 this.set_expiry = function(t_ms){
```

```
140
141     /* follow clock or key to keep the show going */
142     this.expiry_ms = t_ms
143
144     /* state expires by key press */
145     if(t_ms <= 0){
146       this.key_expiry = true
147     }
148   }
149
150   /* enter a state (begin) */
151   this.start = function(){
152     var ctx = get_ctx()
153
154     if(this == ctx.last_state){
155
156         /* go through all the states and record (in string format) the contents, as we'd like it to appear
157         var state_i = ctx.first_state, state_index = 0
158         var message = "url,event_id,task_id,task_type,trial_id,duration(mS),start(yyyy:mm:dd:hh:mn:ss:mls),
159         var pi;
160         for(var state_i = ctx.first_state; state_i != ctx.last_state; state_i = state_i.successor){
161           var stim_type = null;
162           var my_stim  = null;
163
164           /* the right way to check if a variable is undefined or not */
165           if(typeof state_i.pool_id !== 'undefined'){
166             pi = JSON.parse(JSON.stringify(state_i.pool_id))
167           }else{
168             pi = ""
169           }
170
171           if(state_i.wrd_stim){
172             stim_type = "word"
173             my_stim = state_i.wrd_stim
174           }
175
176           if(state_i.img_stim){
177             stim_type = "image"
178             my_stim = state_i.img_stim.fn
179           }
180
181           if(stim_type){
182           }else{
183             stim_type = ""
184           }
185
186           if(my_stim){
187           }else{
188             my_stim = ""
189           }
190
191           /* for a given "state", record a line of data */
192           message += window.location.href.toString() + ","
193           message += state_index.toString() + ","       /* event_id: global index / line number */
194           message += state_i.task_id + ","              /* task_id */
195           message += state_i.type + ","                 /* task_type */
196           message += state_i.trial_id + ","             /* trial_id */
197           message +=  Math.round(10. * (state_i.t1 -  state_i.t0)) / 10. + ","
198           message += parse_date_time(state_i.start_date_time).toString() + ","
199           message += parse_date_time(state_i.end_date_time).toString() + ","
200           if(state_i.type == 'isi'){
201             message += state_i.expiry_ms.toString()
202           }
203           message += ","                                /* ISI */
204           message += ","                                /* SET */
```

```
205           message += stim_type.toString() + ","          /* stim_type */
206           message += my_stim.toString() + ","            /* stim_id */
207           message += pi.toString() + ","                 /* stimulus-pool id */
208           var response = ""
209           for(var k in state_i.key_strokes){
210             response += String.fromCharCode(state_i.key_strokes[k])
211           }
212           message += response + ""                       /* response */
213
214           /* add a newline character */
215           message += "\n"
216           state_index += 1
217         }
218
219         /* window.location.href == http://domain/memory/examples/test_phase/memory.html */
220         var href = window.location.href
221
222         /* remove last three elements from the array: take the page and navigate to: ../../xml-receive.py =
223         var words = href.split('/')
224         var nwords = words.length
225         var target = words.splice(0, nwords-3).join('/') + '/xml-receive.py'
226
227         /* send the message to the server-side script at URL: target */
228         xml_send(message, target)
229     }
230
231     var ctx = get_ctx()
232
233     /* start the clock.. */
234     this.t0 = window.performance.now()
235     this.start_date_time = date_time()
236
237     /* clear the timer */
238     ctx.clear_tmr()
239
240     /* plot the current trial */
241     this.show(ctx)
242
243     /* start the timer? */
244     if(this.expiry_ms > 0){
245       ctx.init_tmr(this.expiry_ms, this.expire)
246     }
247     return null
248   }
249
250   /* pr0c33d t0 th3 n3xt 5+4t3 */
251   this.expire = function(){
252     var ctx = get_ctx()
253
254     /* st0p 4ll th3 cl0ck5 */
255     ctx.clear_tmr()
256
257     /* r3c0rd st0p t1m3 */
258     this.end_date_time = date_time()
259     this.t1 = window.performance.now()
260     var txt = this.txt, suc_txt = null, suc = this.successor
261
262     if(suc!=null && suc.txt !=null){
263       suc_txt = suc.txt
264     }
265
266     /* enter next state */
267     if(this.successor!=null){
268       ctx.set_state(this.successor)
269       ctx.get_state().start()
```

```
270
271       /* this condition might only be good if we have the "score card"? not sure. Replace score card with t
272       if(this.successor.successor == null){
273       }
274     }
275     /* record data to csv-line record (global) here..? */
276
277   }
278   return this
279 }
```

4.7. task.js.

```
1  /* Event hierarchy: 1) Experiment (includes multiple tasks) 2) Task (includes multiple trials) 3) Trial (ea
2
3  /* instructions task (show a slide with a message on it) */
4  function instructions(txt){
5    var my_task_id = next_task_id++
6
7    /* initialize generic "trial" object */
8    var x = new state()
9
10   /* set associated text field */
11   x.txt = txt
12
13   /* no timer for the trial */
14   x.set_expiry(0)
15   x.type = 'instructions'
16   x.task_id = my_task_id
17   x.trial_id = 0
18   return x
19 }
20
21 /* study phase, formerly known as orientation task: multiple 'trials' / events occur here.. random selectio
22 function study_phase(my_pool, isi=0){
23   var my_pools = []
24   if(my_pool.is_pool){
25     my_pools.push(my_pool)
26   }else{
27     my_pools = my_pool
28   }
29
30   var trial_index = -1
31   var my_task_id = next_task_id++
32
33   /* record references to graphics context, and stimulus pool */
34   this.ctx = ctx
35   this.p = my_pools
36   this.pool_ids = new Array()
37
38   for(var a_pool in my_pools){
39     var my_pool = my_pools[a_pool]
40     this.pool_ids.push(my_pool.pool_id)
41
42     /* iterate over selected elements of pool */
43     for(var i  in my_pool.selection_n){
44       trial_index ++
45
46       /* if ISI was set, prefix with a "blank" slide */
47       if(isi > 0){
48         var x = new state()
49         x.set_expiry(isi)
50         x.type = 'isi'
51         x.wrd_stim = ""
```

```
52          x.trial_id = trial_index
53          x.task_id = my_task_id
54          x.set_pool_id(my_pool.pool_id)
55          x.clear_admissible_keys()
56          x.key_expiry = false
57        }
58
59        /* initialize generic "trial" object for each case */
60        var x = new state()
61
62        /* need to add timed parameter to front-end API */
63        x.set_expiry(0)
64
65        /* data (word or image) assigned to "trial" */
66        var data = my_pool.selection_n[i]
67
68        /* discern by image or word, respectively */
69        if( typeof(data) === 'object'){
70          x.img_stim = data
71        }else if(typeof(data) === 'string'){
72          x.wrd_stim = data
73        }
74        x.type = 'study_phase'
75        x.trial_id = trial_index
76        x.task_id = my_task_id
77        x.set_pool_id(my_pool.pool_id)
78
79        /* the ASPECT about set_expiry/ key_expiry needs to go here.. */
80        /* ... */
81
82      } /* for var i  in my_pool.selection_n */
83    } /* for var a_pool in my_pools */
84
85    return this
86  }
87
88  /* test phase, formerly known as recognition task - for this phase, the random selection is shuffled back i
89  function test_phase(my_pool, isi=0){
90    var my_pools = []
91    if(my_pool.is_pool){
92      my_pools.push(my_pool)
93    }else{
94      my_pools = my_pool
95    }
96
97    var trial_index = -1
98    var my_task_id = next_task_id++
99
100   this.ctx = ctx
101   this.p = my_pools
102   this.pool_ids = new Array()
103
104   for(var a_pool in my_pools){
105     var my_pool = my_pools[a_pool]
106     this.pool_ids.push(my_pool.pool_id)
107
108     var trial_index = -1, shuffled_data = my_pool.reshuffle(), shuffled = shuffled_data[0], deja_vu = shuff
109     for(var i in shuffled){
110       trial_index ++
111
112       /* if ISI was set, prefix with a "blank" slide */
113       if(isi > 0){
114         var x = new state()
115         x.set_expiry(isi)
116         x.type = 'isi'
```

```
117         x.wrd_stim = ""
118         x.trial_id = trial_index
119         x.task_id = my_task_id
120         x.set_pool_id(my_pool.pool_id)
121         x.clear_admissible_keys()
122         x.key_expiry = false
123       }
124
125       var x = new state()
126       x.set_expiry(0)
127       x.key_required = true
128       var data = shuffled[i], deja = deja_vu[i]
129
130       /* record within the object: do we have deja-vu? */
131       x.deja = deja
132
133       /* word or image? */
134       if( typeof(data) === 'object'){
135         x.img_stim = data
136       }else if(typeof(data) ==='string'){
137         x.wrd_stim = data
138       }
139       x.type = 'test_phase'
140       x.trial_id = trial_index
141       x.task_id = my_task_id
142       x.set_pool_id(my_pool.pool_id)
143     }
144   }
145   var m = 'Thank you for completing this section. '
146   var end = instructions(m)
147
148   end.action = function(me){
149     var msg = m + 'Your score: ' + ctx.questions_correct.toString() + '/' + ctx.questions_total.toString()
150     me.txt = msg
151   }
152   return this
153 }
154
155 /* previously known as feedback task */
156 function feedback(txt, keys){
157   var my_task_id = next_task_id++
158
159   var x = new state()
160   x.set_expiry(0)
161   x.txt = txt
162   x.key_required = true
163   x.clear_admissible_keys()
164   for(var i in keys){
165     x.add_admissible_key(keys[i])
166   }
167   x.type = 'feedback'
168   x.trial_id = 0
169   x.task_id = my_task_id
170 }
171
172 /* list as many countries as possible during e.g., a 3-minute period (default, 30s) */
173 function delay_task(txt, delay_time=30000){
174   var my_task_id = next_task_id++
175
176   var y = instructions(txt)
177   y.key_expiry = true
178   y.set_expiry(500)
179
180   /* keypress activated with minimum time */
181   y.hold_on()
```

```
182
183   /* time [mS] */
184   var thirty_seconds = 30000, x = new state()
185   x.set_expiry(delay_time)
186   x.key_expiry = false
187   x.txt = ''
188   x.type = 'delay'
189   x.trial_id = 0
190   x.task_id = my_task_id
191   return this;
192 }
```

4.8. text.js.

```
1  /* wrap text around a window region-- via ashblue */
2  function wrap_text(s, ctx, start_y=0){
3    var myX = 10, myY = 50, line = '', lines = [], w = ctx.w(), h = ctx.h(), line_test = '', words = s.split(
4    ctx.font = font_size +'px Arial'
5
6    /* place words one by one */
7    for(var j = 0; j < words.length; j++){
8      line_test = line + words[j] + ' '
9
10     /* wrap if over the edge */
11     if(ctx.measureText(line_test).width > w){
12       myY = lines.length * font_size + font_size
13       lines.push({text: line, height: myY})
14       line = words[j] + ' '
15     }else{
16       line = line_test
17     }
18   }
19
20   /* catch last line if something left over */
21   if(line.length > 0){
22     current_y = lines.length * font_size + font_size
23     lines.push({text: line.trim(), height: current_y})
24   }
25
26   /* plot text */
27   for(var j = 0, len = lines.length; j < len; j++){
28     ctx.fillText(lines[j].text, 0, lines[j].height + start_y)
29   }
30 }
31
32 function centre_text(s){
33   var font_size = ctx.font_size, textString = s;
34   ctx.font = 30 +'px Arial'
35   textWidth = ctx.measureText(textString ).width
36   ctx.fillText(textString, (canvas.width/2) - (textWidth / 2), canvas.height/2)
37 }
```

4.9. util.js.

```
1  /* get date and time */
2  function date_time(){
3    return new Date()
4  }
5
6  /* seed for rand() below */
7  var seed = 5
8
9  /*random-number generator http://indiegamr.com/generate-repeatable-random-numbers-in-js/ : initial seed.. i
10 function rand(max, min) {
```

```
  max = max || 1
  min = min || 0
  seed = (seed * 9301 + 49297) % 233280
  var rnd = seed / 233280
  return min + rnd * (max - min)
}

/* pad to length n (with 0's on the left) */
function pad_n(x, n){
  var s = parseInt(trim(x)).toString(), m = s.length, d = n - m
  if(d > 0){
    s += '0'.repeat(d)
  }
  return s
}

/* via stackoverflow.com/users/4321/jw */
function get_keys(dictionary){

  /* keys recursive */
  var keys = []

  /* filter for direct ancestors */
  for(var key in dictionary){
    if(dictionary.hasOwnProperty(key)){
      keys.push(key)
    }
  }
  return keys
}

/* draw an image */
function draw_img(x, ctx){
    var cf = 4 * ctx.font_size, h = ctx.h() - cf, w = ctx.w(), lw = x.width, lh = x.height, sf = Math.min(w
(-20 + cf / 2)
    ctx.drawImage(x, a, b + df, c, d)
}

/* write the above to a standardized format */
function parse_date_time(today){

  /* most significant units first */
  var bits = [today.getFullYear(), today.getMonth() + 1, today.getDate(), today.getHours(), today.getMinute

  /* pad with zeros */
  for(var i = 0; i < bits.length; i++){
    var n_pad = 2
    if(i==0) n_pad = 4
    if(i== 6) n_pad = 3
    var bts = bits[i].toString()
    bits[i] = pad_n(bts, n_pad)
  }
  return(bits.join(':'))
}

/* "faster trim" via blog.stevenlevithan.com */
function trim(s){
  return s.toString().replace(/^\s\s*/,'').replace(/\s\s*$/,'')
}

/* send text format data (string s) via XML to receive script at url (string): xml-receive_script_url
*/
function xml_send(s, xml_receive_script_url){

  /* xml http request object */
```

```
74   var xhr = (window.XMLHttpRequest) ? new XMLHttpRequest() : new activeXObject("Microsoft.XMLHTTP")
75   var data = new FormData()
76   data.append("data", s)
77   xhr.open( 'post', xml_receive_script_url, true)
78   xhr.send(data)
79 }
```

5. Source Code: Server Side

The folder data/ in the directory structure: if it doesn't yet exist, the server-side python code will create it.

5.1. xml-receive.py.

```
1 #!/usr/bin/python
2 ''' server-side python-CGI script to receive text data sent over
3 the internet by the client-side function util.js::xml_send()'''
4 import os
5 import cgi
6 import uuid
7 import datetime
8
9 # create /data folder if it does not yet exist
10 dat_f = os.getcwd() + '/data/'
11 if not os.path.exists(dat_f):
12     os.mkdir(dat_f)
13
14 # retrieve CGI form data
15 dat = None
16 try:
17     dat = str(cgi.FieldStorage().getvalue('data'))
18 except:
19     pass
20
21 # write the data to file in the data/ folder
22 if dat:
23     fn = dat_f + str(datetime.datetime.now().isoformat())
24     open(fn + '_' + str(uuid.uuid4().hex) + '.txt', 'wb').write(dat)
```

6. Recommendations For Further Improvements

Here's a short point-form list of possible improvements to the software:

- Finish drag-and drop implementation, that
  - does not allow invalid experiments to be constructed
  - removes any technicality from the process of coding an experiment
- Smarter image loading
  - Only load the images that are actually used in the experiment
  - Automagically detect available images from folder