

# The AcroMemory Package

## A member of the AeB Pro family

D. P. Story  
Email: `storyd@owc.edu`

processed June 26, 2020

## Contents

<b>1 Introduction</b>	<b>1</b>
<b>2 Creating the Image Tiles</b>	<b>2</b>
<b>3 Main Macro Code</b>	<b>3</b>
<b>4 Document JavaScript for AcroMemory</b>	<b>9</b>
<b>5 Index</b>	<b>17</b>
<b>6 Change History</b>	<b>18</b>

## 1 Introduction

At the instigation of my erstwhile friend, Jüergen, I present to you AcroMemory, and for the life of me, I can't remember why.

Oh, yes, AcroMemory is a memory game in which you find the matching tiles. There are two versions —available as options of this package—for your enjoyment, `acromemory1` and `acromemory2` (the default).

- `acromemory1`: Here you have a single game board, a rectangular region divided by rows and columns. The total number of tiles should be even, each tile should have a matching twin. The game begins with all the tiles hidden. the user clicks a tile, then another. If the tiles do not match, they become become hidden again (you did remember the position of those tiles, didn't you?); otherwise, they remain visible and are now read-only. The game is complete when the user, with a lot of time on his/her hands, matches all tiles. There is a running tabulation kept on the number of tries. There is also a button which resets the game and randomizes the tiles.

- **acromemory2**: For this game you have two identical rectangular images subdivided into tiles (or slices) arrayed in rows and columns. The tiles for one of the two images has been randomly re-arranged. The object of the game is to find all the matching tiles by choosing a tiles from one image, and tile from the other image. As in the first case, if the selected tiles do not match, they are hidden after an short interval of time (you did remember the position of those tiles, didn't you?); otherwise, they remain visible and are now read-only. The game is over when all tiles are matched, when this occurs, end-of-game special effects occur that will dazzle the senses. There is an option to view a small image to help you locate the matching tiles on the non-randomized; useful if the image is complex. There is no reset button at this time, to play again, the user must close and open the document.

The demo files are `acromemory1.tex` and `acromemory2.tex`. These files show how to lay out the various elements of this package.

**What's New for version v2.0 (2020-06-23)**: Rewrote the entire package to support all L<sup>A</sup>T<sub>E</sub>X workflows: pdf<sub>l</sub>atex, lua<sub>l</sub>atex, xel<sub>l</sub>atex, and dvips -> distiller.

## 2 Creating the Image Tiles

For `acromemory2`, slicing of the image is at the very heart of this game. You can slice an image in to rectangular tiles using any of several applications: Adobe Illustrator, Photoshop and ImageReady, for example. But these are expensive applications; a cheap method is to use the L<sup>A</sup>T<sub>E</sub>X package `tile-graphic`.<sup>1</sup>

```

1 <*package>
2 \RequirePackage{xkeyval}

acromemory1 One playing board, where you try to match identical icons.
3 \DeclareOptionX{acromemory1}{\acromemoryitruel}

acromemory2 Two playing boards, one board randomized the other not. Try to find the match-
ing icons, one from each of the two boards.
4 \DeclareOptionX{acromemory2}{\acromemoryifalse}

includehelp Only valid when acromemory2 is taken, this option allows you to provide a figure
showing the completed puzzle.
5 \DeclareOptionX{includehelp}{\includehelptrue}

draft Draft mode, works for pdflatex and lualatex only.
6 \DeclareOptionX{draft}{\PassOptionsToPackage{draft}{graphicx}}

Declare to booleans, and process options
7 \newif\ifincludehelp \includehelpfalse
8 \newif\ifacromemoryi \acromemoryifalse
9 \ProcessOptionsX\relax
10 \@ifpackageloaded{eforms}{\execJSOn}

```

<sup>1</sup><https://ctan.org/pkg/tile-graphic>

```

11  {\RequirePackage[execJS]{eforms}}
12  \RequirePackage{aeb-comment}
13  \ifxetex\makeXasPDOff\fi
14  \RequirePackage{icon-appr}
15  \RequirePackage{multido}
16  \RequirePackage{graphicx}
17  \ifacromemoryi
18    \def\RanIdentifier{\@gobble}
19    \includecomment{acromemory1}
20    \excludecomment{acromemory2}
21    \includehelpfalse
22  \else
23    \def\RanIdentifier{R\@gobble}
24    \includecomment{acromemory2}
25    \excludecomment{acromemory1}
26  \fi
27  \newcount\am@nCnt

```

### 3 Main Macro Code

`\bDebug` A debugging command. When executed in the preamble, more is written to the Acrobat console as the document is opened the first time, also, the icons are initially visible so you can see the layout, and quickly play the game. This was used in development extensively to help develop the JavaScript.

```

28  \def\bDebug{\def\memDebug{true}}
29  \def\memDebug{false}

```

`\isPackage` Placed prior to `\amEmbedTiles`, it signals that the images are in a package file.

```

30  \newif\if@isPackaged\@isPackagedfalse
31  \def\isPackage{\@isPackagedtrue}
32  \let\amIconObjs\@gobble

```

`\amEmbedTiles[(ext)]{(name)}{(n-rows)}{(n-cols)}{(path)}` Embed the required files for this puzzle. We require a *(name)*, in the off chance that some day more than one puzzles are allowed.

```

33  \newcommand{\amEmbedTiles}[4][\begingroup
34    \gdef\amNumImages{#3}%
35    \csarg\gdef{amGraphicPath#2}{#4}%
36    \gdef\imageImportPath{#4}%
37    \ifacromemoryi

```

If `acromemoryi` is in effect, then `#3` is half the required icons, each icon is placed twice, then mixed up. Anyway, we double this value going forward.

```

38    \@tempcnta=#3\relax
39    \multiply\@tempcnta\tw@
40    \xdef\nTotalTiles{\the\@tempcnta}\else
41    \gdef\nTotalTiles{#3}\fi
42  \def\@Ext{#1}\ifx\@Ext\@empty\def\@Ext{.pdf}\else\def\@Ext{.#1}\fi
43  \@tempcnta\z@

```

```

44 \let\@embedList\@empty
45 \let\AMIndxList\@gobble
46 \edef\z{\noexpand\g@addto@macro\noexpand
47   \amIconObjs{"#2":\amNumImages}}\z
48 \@whilenum \@tempcnta < \amNumImages \do{%
49   \am@nCnt\@tempcnta\advance\am@nCnt\@ne
50   \ifnum\am@nCnt<10\relax\edef\x{0\the\am@nCnt}\else
51     \edef\x{\the\am@nCnt}\fi
52   \edef\z{\noexpand\g@addto@macro\noexpand\AMIndxList{"#2pic\x"}}\z
53   \ifxetex\if@isPackaged
54     \PackageWarning{acromemory}
55     {There is no support for embedding packaged\MessageBreak
56     PDFs with xelatex. Ignoring the \string\isPackage\MessageBreak
57     command}%
58     \@isPackagedfalse
59   \fi\fi
60   \ifacromemoryi
61     \@tempcntb\@tempcnta
62     \multiply\@tempcntb\tw@
63     \advance\@tempcntb\@ne
64     \edef\z{\the\@tempcntb}\advance\@tempcntb\@ne
65     \edef\zi{\the\@tempcntb}%
66     \if@isPackaged
67       \ifpdf
68         \edef\y{\noexpand
69           \embedIcon[name=#2pic\x,%
70             hyopts={page=\x}]{#4_package.pdf}}%
71       \else
72         \edef\y{\noexpand
73           \embedIcon[name=#2pic\x,%
74             placement={ [1]Membutton.\z, [1]Membutton.\zi},%
75             page=\x-1}{#4_package.pdf}}%
76       \fi
77     \else
78       \edef\y{\noexpand
79         \embedIcon[name=#2pic\x,%
80           placement={ [1]Membutton.\z, [1]Membutton.\zi}]{#4_\x\@Ext}}%
81       \fi
82     \else
83       \ifincludehelp\embedIcon[name=helpimage,%
84         placement={ [1]memoryhelp}]{#4\@Ext}\fi
85     \edef\z{\the\am@nCnt}%
86     \if@isPackaged
87       \ifpdf
88         \edef\y{\noexpand
89           \embedIcon[name=#2pic\x,%
90             hyopts={page=\x}]{#4_package.pdf}}%
91       \else
92         \edef\y{\noexpand
93           \embedIcon[name=#2pic\x,%

```

```

94         placement={ [1]MemLbutton.\z, [1]MemRbutton.\z},%
95         page=\x-1]{#4_package.pdf}}%
96     \fi
97     \else
98         \edef\y{\noexpand
99         \embedIcon[name=#2pic\x,%
100         placement={ [1]MemLbutton.\z, [1]MemRbutton.\z}%
101         ]{#4_\x\@Ext}}%
102     \fi
103     \fi
104     \expandafter\g@addto@macro\expandafter
105     \@embedList\expandafter{y}%
106     \@tempcnta\am@nCnt
107 }% do
108 \toks@=\expandafter{\@embedList}\the\toks@
109 %\typeout{!! \the\toks@}%
110 \endgroup
111 \global\@isPackagedfalse
112 }

```

`\amIconPic[<opts>]{<fname>}{<wd>}{<ht>}` A general purpose push button that will have icon appearances.

```

113 \newcommand{\amIconPic}[4] [] {% \I{csOf{name}} required
114 \pushButton[\BG]{\W{1}\S{S}#1\TP{1}\%F{FHidden}
115 ]{#2}{#3}{#4}}

```

`\insertTiles{<name>}{<width>}{<rows>}{<cols>}` Command for placing the tiles of a picture. We assume that the pictures are numbered consecutively across rows.

*<name>* The name of the graphic (a JavaScript identifier)  
*<width>* The width of the image, the height is scaled proportionally  
*<rows>* The number of rows  
*<cols>* The number of columns

```

116 \newcommand\insertTiles[4]{\begingroup
117 \@tempdima#2\relax
118 \divide\@tempdima #4\relax
119 \setbox\z@\hbox{\includegraphics[draft,width=\@tempdima]%
120 {\@nameuse{amGraphicPath#1}}}%
121 \edef\amTileWd{\the\wd\z@}%
122 \setlength\@tempdima{\ht\z@+\dp\z@}%
123 \setbox\z@\box\voidb@x
124 \edef\amTileHt{\the\@tempdima}%
125 \@tempdima\amTileWd\relax
126 \multiply\@tempdima #4\relax
127 \edef\tot@lWd{\the\@tempdima}%
128 \@tempcnta#3\relax
129 \multiply\@tempcnta #4\relax
130 \divide\@tempcnta\tw@
131 \edef\tot@lHalfTiles{\the\@tempcnta}%

```

```

132 \begin{minipage}{\tot@lWd}%
133   \offinterlineskip\hbadness=10000\@tempcnta\z@
134   \leavevmode
135   \rlap{\amIconPic[\BC{}BG{}]{nullIconBtn}{0bp}{0bp}}%
136   \multido{\i=1+1}{\tot@lHalfTiles}{%
137     \advance\@tempcnta\@ne
138     \edef\y{\the\@tempcnta}%
139     \ifnum\i<10\relax
140       \edef\x{0\i}\else
141       \edef\x{\i}\fi
142     \edef\iconPresets{\noexpand\IX{\noexpand\csOf{#1pic\x}}}%
143     \amIconPic[\AAmouseup{selectTile();}\FB{true}
144       \presets{\iconPresets}\presets{\amtile@KVs}
145     ]{Membutton.\y}{\amTileWd}{\amTileHt}\allowbreak
146     \advance\@tempcnta\@ne
147     \edef\y{\the\@tempcnta}%
148     \edef\iconPresets{\noexpand\IX{\noexpand\csOf{#1pic\x}}}%
149     \amIconPic[\AAmouseup{selectTile();}\FB{true}
150       \presets{\iconPresets}\presets{\amtile@KVs}
151     ]{Membutton.\y}{\amTileWd}{\amTileHt}\allowbreak
152   }% multido
153 \end{minipage}%
154 \endgroup
155 }
156 \def\amtileKVs#1{\def\amtile@KVs{#1}}
157 \amtileKVs{}

```

`\insertTilesii{<name>}{<width>}{<n-rows>}{<n-cols>}{<L|R>}`

- `<name>` The name of the graphic (a JavaScript identifier)
- `<width>` The width of the image, the height is scaled proportionally
- `<rows>` The number of rows
- `<cols>` The number of columns
- `<L|R>` Indicates for Left or Right Image

Is the common code for `\insertTilesL` and `\insertTilesR`.

```

158 \newcommand\insertTilesii[5]{\begingroup
159   \def\@rgv{#5}\def\as@L{L}%
160   \@tempdima#2\relax
161   \setbox\z@\hbox{\includegraphics[draft,width=\@tempdima]{%
162     \@nameuse{amGraphicPath#1}}}%
163   \edef\amImageWd{\the\wd\z@}%
164   \setlength\@tempdima{\ht\z@+\dp\z@}%
165   \setbox\z@\box\voidb@x
166   \edef\amImageHt{\the\@tempdima}%
167 % Now calculate wd and ht of a tile
168   \@tempdima\amImageWd\relax
169   \divide\@tempdima#4\relax
170   \edef\amTileWd{\the\@tempdima}%

```

```

171 \@tempdima\amImageHt\relax
172 \divide\@tempdima#3\relax
173 \edef\amTileHt{\the\@tempdima}%
174 % Calculate total number of tiles
175 \@tempcnta#3\relax
176 \multiply\@tempcnta#4\relax
177 \edef\Tot@lTiles{\the\@tempcnta}%
178 % Begin minipage of width \amImageWd
179 \begin{minipage}{\amImageWd}%
180   \offinterlineskip\hbadness=10000\@tempcnta\z@
181   \leavevmode
182   \rlap{\amIconPic[\BC{}BG{}]{nullIconBtn}{0bp}{0bp}}%
183   \multido{\i=1+1}{\Tot@lTiles}{%
184     \advance\@tempcnta@one
185     \edef\y{\the\@tempcnta}%
186     \ifnum\i<10\relax
187       \edef\x{0\i}\else
188       \edef\x{\i}\fi
189     \ifx\@rgv\as@L
190       \def\muAction{nRowsAM=#3;nColsAM=#4;\string\r
191         selectNonRandomTile(\y,\y);} \else
192       \def\muAction{nRowsAM=#3;nColsAM=#4;\string\r
193         selectRandomTile(randomAM[\y],\y);} \fi
194     \edef\iconPresets{\noexpand\AAMouseup{\muAction}\noexpand
195       \IX{\noexpand\csOf{#1pic\x}}}%
196     \amIconPic[\presets{\iconPresets} %\FB{true}
197       \presets{\amtile@KVs}
198     ]{Mem#5button.\y}{\amTileWd}{\amTileHt}\allowbreak
199   }% multido
200 \end{minipage}\endgroup
201 }

```

`\insertTilesL{<name>}{<width>}{<n-rows>}{<n-cols>}` Inserts the left-hand tiles, which is the non-randomize version of the picture.

```

202 \newcommand\insertTilesL[4]{\ifacromemoryi
203   \def\AM@next{\PackageWarning{acromemory}
204     {The use of \string\insertTilesL\space is supported\MessageBreak
205     only for the acromemory2 option}}\else
206   \def\AM@next{\insertTilesii{#1}{#2}{#3}{#4}{L}}\fi\AM@next}

```

`\insertTilesR{<name>}{<width>}{<n-rows>}{<n-cols>}` Inserts the right-hand tiles, which is the randomized version of the picture.

```

207 \newcommand\insertTilesR[4]{\ifacromemoryi
208   \def\AM@next{\PackageWarning{acromemory}
209     {The use of \string\insertTilesR\space is supported\MessageBreak
210     only for the acromemory2 option}}\else
211   \def\AM@next{\insertTilesii{#1}{#2}{#3}{#4}{R}}\fi\AM@next}

```

`\helpImage[<eform-opts>]{<width>}` When `acromemory2` option and the `includehelp` option are taken, these commands are available. The command `\helpImage` will contain

an icon of the puzzle, and its width is set by the command `\setHelpImageWidth`. The image is normally hidden until the user rolls over the `\rolloverHelpButton`. The icon appears with a caption under it, the content of the caption can be entered using `\theHelpCaption`.

```

212 \newcommand{\helpImage}[2] []{%
213     \ifincludehelp{\setbox\z@\hbox{%
214         \includegraphics[draft,width=#2]{\imageImportPath}}%
215     \dimen\z@=\ht\z@\advance\dimen\z@14bp\ht\z@=\dimen\z@
216     \pushButton[\IX{\csOf{helpimage}}\TP{2} %\CA{\helpCaption}
217     \Ff\FfReadOnly\BC{} \BG{} \S{S}#1]%
218     {memoryhelp}{\the\wd\z@}{\the\ht\z@}}\fi
219 }

```

`\rolloverHelpButton`

```

220 \newcommand{\rolloverHelpButton}[3] []{%
221     \ifincludehelp
222         \pushButton[\CA{Help}\BC{0 0 1}\BG{0.89 0.9 0.9}
223         \AA{\AAMouseEnter{\JS{%
224             var f = this.getField("memoryhelp");\r
225             oIcon = f.buttonGetIcon(1);\r
226             f.buttonPosition = position.iconTextV;\r
227             f.buttonSetIcon(oIcon,0);\r
228             f.buttonSetCaption({cCaption: "\helpCaption"});\r
229             f.textColor=color.blue;\r
230             }}%
231         \AAMouseExit{\JS{%
232             var f = this.getField("memoryhelp");\r
233             f.buttonPosition = position.iconOnly;\r
234             f.buttonSetIcon(nullIcon,0);
235             }}}#1]{checkhelp}{#2}{#3}%
236     \fi
237 }

```

`\theHelpCaption`

```

238 \def\theHelpCaption#1{\def\helpCaption{#1}}
239 \theHelpCaption{A little help}

```

`\messageBox[<opts>]{<wd>}{<ht>}` A message text field, as the user works the puzzle, the progress is reported to this field.

```

240 \newcommand{\messageBox}[3] []{%
241     \textField[#1\Ff\FfMultiline]{MsgBox}{#2}{#3}}

```

`\playItAgain[<opts>]{<wd>}{<ht>}` For the `acromemory2` option, this button can be placed to reset the two memory boards, so the memory game can be played again.

```

242 \newcommand{\playItAgain}[3] []{\ifacromemoryi
243     \pushButton[\CA{Play again}#1\AAMouseup{playagain();}]%
244     {playAgain}{#2}{#3}\fi
245 }

```



`\playItAgain[(opts)]{(wd)}{(ht)}` For the `acromemory1` option, this button can be placed to reset the game board, the icons are rearranged and hidden again.

```
246 \newcommand{\tryItAgain}[3][\ifacromemoryi\else
247   \pushButton[\CA{Test Your Memory}#1\AAmouseup{tryAgain();}]%
248   {testYourMemory}{#2}{#3}\fi
249 }
```

## 4 Document JavaScript for AcroMemory

Operational support is provided by JavaScript.

```
250 \newcommand{\initFirstMsg}{Press the 'Play again'
251   button to initialize the puzzle"}
252 \newcommand{\initFirstMsg}{Press the 'Test Your Memory'
253   button to initialize the puzzle"}
254 \begin{insDLJS*}{memjs}
255 \begin{newsegment}{AcroMemory 1: Global Data and Initialization}
256 // Global Data:
257 var isRandomized=false;
258 var randomAM = new Array(\nTotalTiles+1);
259 var imageNames = new Array(\AMIndxList);
260 imageNames.push(\AMIndxList);
261 imageNames.unshift("null");
262 var dpsl = randomAM.length;
263 var timeout = 10;
264 var shutdown, rAE;
265 var ok2Continue = true;
266 var nRowsAM, nColsAM;
267 var nCorrectAM = 0;
268 var nAttemptsAM = 0;
269 for (i=1; i<=\nTotalTiles; i++) randomAM[i]=i;
```

We get the push button with a null icon (`nullIconBtn`) We get the null icon object from it. This technique eliminates the previous need for the Acrobat application when viewing the game.

```
270 var f=this.getField("nullIconBtn");
271 var nullIcon=f.buttonGetIcon();
272 var debug = \memDebug;
273 \end{newsegment}
274 \begin{acromemory1}
275 \begin{newsegment}{AcroMemory 2: Initialize Pic Names}
276 var currentChoice = "";
277 var currentIconName = "";
278 \end{newsegment}
279 \end{acromemory1}
280 \begin{acromemory2}
281 \begin{newsegment}{AcroMemory 2: Initialize Pic Names}
282 var LcurrentChoice = 0;
283 var LcurrentTile = 0;
```

```

284 var RcurrentChoice = 0;
285 var RcurrentTile = 0;
286 \end{newsegment}
287 \end{acromemory2}
288 \begin{newsegment}{AcroMemory 3: Bubble Sort}
289 function clearAM()
290 {
291     for ( var i=1; i<=\nTotalTiles; i++ )
292     {
293         var f = this.getField("Mem\RanIdentifier button."+i);
294         f.buttonSetIcon(nullIcon);
295     }
296 }
297 function mixupAM()
298 {
299     var i, rand;
300     for (i=1; i<= \nTotalTiles; i++)
301     {
302         var rand = Math.random();
303         rand *= dpsl*dpsl;
304         rand = Math.ceil(rand);
305         rand = rand \% dpsl;
306         if (rand == 0 ) rand = 1;
307         temp = randomAM[i];
308         randomAM[i]=randomAM[rand];
309         randomAM[rand]=temp;
310     }
311 }
312 function showAM()
313 {
314     for ( var i=1; i<=\nTotalTiles; i++ )
315     {
316         var oIcon = this.getIcon(imageNames[randomAM[i]]);
317         var f = this.getField("Mem\RanIdentifier button."+i);
318         f.buttonSetIcon(oIcon);
319     }
320 }
321 // Begin bubble sort
322 function sortoutAM()
323 {
324     outerLoop(randomAM.length-1);
325 }
326 function outerLoop(i)
327 {
328     if ( ok2Continue && (i >= 0) ) shutdown = %
329 app.setTimeout("app.clearTimeout(shutdown); %
330 innerLoop("+i+",1);", timeout);
331 }
332 function innerLoop(i,j)
333 {

```

```

334     if ( j <= i )
335     {
336         if (randomAM[j-1] > randomAM[j])
337         {
338             var temp = randomAM[j-1];
339             randomAM[j-1] = randomAM[j];
340             randomAM[j] = temp;
341             var oIcon = this.getIcon(imageNames[randomAM[j-1]]);
342             var f = this.getField("Mem\RanIdentifier button."+(j-1));
343             f.buttonSetIcon(oIcon);
344             var oIcon = this.getIcon(imageNames[randomAM[j]]);
345             var f = this.getField("Mem\RanIdentifier button."+j);
346             f.buttonSetIcon(oIcon);
347         }
348         j++;
349         if ( ok2Continue ) shutdown = %
350 app.setTimeout("app.clearTimeout(shutdown); %
351 innerLoop("+i+", "+j+", "", timeout);
352     }
353     else
354     {
355         i--;
356         outerLoop(i);
357     }
358 }
359 function randomizePuzzle() {
360     mixupAM();
361     for ( var i=1; i<=\nTotalTiles; i++) {
362         var g = this.getField("Mem\RanIdentifier button."+i);
363         var oIcon = this.getIcon(imageNames[randomAM[i]]);
364         g.buttonSetIcon(oIcon,1);
365         if (debug) g.buttonSetIcon(oIcon,0);
366     }
367     isRandomized=true;
368 }
369 \end{newsegment}
370 \begin{acromemory1}
371 \begin{newsegment}{AcroMemory 4: Tile Processing}
372 var currentIndex="";
373 var currentName="";
374 var _bOK1=true;
375 function selectTile() // right side randomly arranged
376 {
377     if(!isRandomized){
378         app.alert(\initFirstiMsg);
379         return;
380     }
381     if (!_bOK1) return;
382     var f = event.target;
383     var oIcon = f.buttonGetIcon(1);

```

```

384     f.buttonSetIcon(oIcon,0);
385     var fname = f.name;
386     var re1 = /Membutton\.(\\d+)/;
387     var index = re1.exec(fname);
388     if (debug) console.println("index = " + index[1]);
389     var thisiconName = imageNames[randomAM[index[1]]];
390     if (debug) console.println("thisiconName = " + thisiconName);
391     if ( currentChoice == "" ) {
392         currentChoice = fname;
393         currentIconName = thisiconName;
394         return;
395     }
396     if ( (thisiconName == currentIconName) )
397     { // right choice
398         nCorrectAM++;
399         nAttemptsAM++
400         f.readonly = true;
401         var g = this.getField(currentChoice);
402         g.readonly = true;
403         reportProgress(nCorrectAM,nAttemptsAM);
404         resetCountersAM();
405
406     } else { // wrong choice
407         nAttemptsAM++
408         _bOK1=false;
409         reportProgress(nCorrectAM,nAttemptsAM);
410         rAE = app.setTimeout(
411 "resetAfterError(\""+currentChoice+"\", \""+fname+"\"");%
412 _bOK1=true;", 1000);
413         resetCountersAM();
414     }
415
416 }
417 function resetCountersAM ()
418 {
419     currentChoice = "";
420     currentIconName = "";
421 }
422 function resetAfterError(l,r)
423 {
424     try { app.clearTimeout(rAE); } catch(e) {};
425     var f = this.getField(l);
426     var g = this.getField(r);
427     if (!debug) g.buttonSetIcon(nullIcon,0);
428     if (!debug) f.buttonSetIcon(nullIcon,0);
429 }
430 function executePostGameEffects() {return;}
431 function playagain()
432 {
433     for ( var i=1; i<=\\nTotalTiles; i++) {

```

```

434         var g = this.getField("Membutton."+i);
435         g.buttonSetIcon(nullIcon,0);
436     }
437     g = this.getField("Membutton");
438     g.readonly=false;
439     resetCountersAM();
440     nCorrectAM = 0;
441     nAttemptsAM = 0;
442     reportProgress(nCorrectAM,nAttemptsAM);
443     randomizePuzzle();
444 }
445 \end{newsegment}
446 \end{acromemory1}
447 \begin{acromemory2}
448 \begin{newsegment}{AcroMemory 4: Tile Processing}
449 // save original positions of fields
450 var aLRect=new Array();
451 var aRRect=new Array();
452 aLRect.push("null");
453 aRRect.push("null");
454 var f=this.getField("MemLbutton");
455 var g=f.getArray();
456 for (var i=0; i<g.length; i++)aLRect.push(g[i].rect);
457 var f=this.getField("MemRbutton");
458 var g=f.getArray();
459 for (var i=0; i<g.length; i++)aRRect.push(g[i].rect);
460 var _bOK2=true;
461 function selectRandomTile(nCnt,n) // right side randomly arranged
462 {
463     if(!isRandomized){
464         app.alert(\initFirstiiMsg);
465         return;
466     }
467     if (!_bOK2) return;
468     if ( RcurrentChoice != 0 ) return;
469     RcurrentChoice = nCnt;
470     RcurrentTile = n;
471     nAttemptsAM++;
472     var f = event.target;
473     f.strokeColor = ["RGB", 0, .6, 0];
474     var oIcon = f.buttonGetIcon(1);
475     f.buttonSetIcon(oIcon,0);
476     if ( LcurrentChoice != 0 ) {
477         if (debug) console.println(%
478 "LcurrentChoice = " + LcurrentChoice + ", RcurrentChoice = " %
479 + RcurrentChoice);
480         if ( LcurrentChoice == nCnt ) { // right answer
481             // need to make right side hidden and readonly
482             // need to make this button readonly

```

```

483         var g = this.getField("MemLbutton."+LcurrentChoice);
484         g.strokeColor=color.transparent;
485         g.readonly = true;
486         f.strokeColor=color.transparent;
487         f.readonly = true;
488         if (++nCorrectAM == \nTotalTiles ) // game complete
489             executePostGameEffects();
490         reportProgress(nCorrectAM,nAttemptsAM);
491         resetCountersAM();
492     } else { // wrong answer
493         // need to set current choices back to zero
494         reportProgress(nCorrectAM,nAttemptsAM);
495         _bOK2=false;
496         rAE = app.setTimeout("resetAfterError(%
497 "+LcurrentTile+", "+RcurrentTile+");_bOK2=true;", 1000);
498         resetCountersAM();
499     }
500 }
501 }

502 // left side, arranged in natural order
503 function selectNonRandomTile(nCnt,n)
504 {
505     if(!isRandomized){
506         app.alert(\initFirstiiMsg);
507         return;
508     }
509     if (!_bOK2) return
510     if ( LcurrentChoice != 0 ) return;
511     LcurrentChoice = nCnt;
512     LcurrentTile = n;
513     var f = event.target;
514     f.strokeColor = ["RGB", 0, .6, 0];
515     var oIcon = f.buttonGetIcon(1);
516     f.buttonSetIcon(oIcon,0);
517     if ( RcurrentChoice != 0 ) {
518         if (debug) console.println(%
519 "LcurrentChoice = " + LcurrentChoice + ", RcurrentChoice = " %
520 + RcurrentChoice);
521         if ( RcurrentChoice == nCnt ) { // right answer
522             // need to make right side hidden and readonly
523             // need to make this button readonly
524             var g = this.getField("MemRbutton."+RcurrentTile);
525             g.strokeColor=color.transparent;
526             g.readonly = true;
527             f.readonly = true;
528             f.strokeColor=color.transparent;
529             if (++nCorrectAM == \nTotalTiles ) // game complete
530                 executePostGameEffects();
531             reportProgress(nCorrectAM,nAttemptsAM);

```

```

532         resetCountersAM();
533     } else { // wrong answer
534         // need to set current choices back to zero
535         reportProgress(nCorrectAM,nAttemptsAM);
536         _bOK2=false;
537         rAE = app.setTimeout(%
538 "resetAfterError("+LcurrentTile+","+RcurrentTile+"");%
539 _bOK2=true;", 1000);
540         resetCountersAM();
541     }
542 }
543 }

544 function resetCountersAM ()
545 {
546     LcurrentChoice = 0;
547     RcurrentChoice = 0;
548     LcurrentTile = 0;
549     RcurrentTile = 0;
550 }

551 function resetAfterError(l,r)
552 {
553     try { app.clearTimeout(rAE); } catch(e) {};
554     var f = this.getField("MemLbutton."+l);
555     var g = this.getField("MemRbutton."+r);
556     if (!debug) g.buttonSetIcon(nullIcon,0);
557     g.strokeColor=color.black;
558     if (!debug) f.buttonSetIcon(nullIcon,0);
559     f.strokeColor=color.black;
560 }

561 function tryAgain() {
562     nCorrectAM=0;
563     nAttemptsAM=0;
564     reportProgress(nCorrectAM,nAttemptsAM);
565     % this.resetForm("MsgBox");
566     resetCountersAM();
567     this.delay=true;
568     for(var i=1; i<=20; i++) {
569         var f=this.getField("MemLbutton."+i);
570         var g=this.getField("MemRbutton."+i);
571         f.buttonSetIcon(nullIcon,0);
572         f.rect=aLRect[i];
573         g.buttonSetIcon(nullIcon,0);
574         g.rect=aRRect[i];
575     }
576     var f=this.getField("MemLbutton");
577     var g=this.getField("MemRbutton");
578     f.lineWidth=1;
579     f.strokeColor=color.black;
580     f.readonly=false;

```

```

581 g.lineWidth=1;
582 g.strokeColor=color.black;
583 g.readonly=false;
584 this.delay=false;
585 randomizePuzzle();
586 }

587 function executePostGameEffects() {
588     sortoutAM();
589     var fL = this.getField("MemLbutton.1");
590     var fR = this.getField("MemRbutton.1");
591     var LulCorner = fL.rect;
592     var RulCorner = fR.rect;
593     var mWidth = LulCorner[2]-LulCorner[0];
594     var mHeight = LulCorner[1]-LulCorner[3];
595     var nCnt = 0;
596     for ( var i=0; i<nRowsAM; i++) {
597         for ( var j=0; j<nColsAM; j++ ) {
598             nCnt++;
599             try {
600                 var g = this.getField("MemLbutton."+nCnt);
601                 g.rect = %
602 [ LulCorner[0]+j*mWidth, LulCorner[1]-i*mHeight, %
603 LulCorner[0]+(j+1)*mWidth, %
604 LulCorner[1]-(i+1)*mHeight ]
605                 g.lineWidth = 0;
606                 g.strokeColor = color.transparent;
607                 var h = this.getField("MemRbutton."+nCnt);
608                 h.rect = [ RulCorner[0]+j*mWidth, %
609 RulCorner[1]-i*mHeight, %
610 RulCorner[0]+(j+1)*mWidth, RulCorner[1]-(i+1)*mHeight ]
611                 h.lineWidth = 0;
612                 h.strokeColor = color.transparent;
613             } catch(e) { %
614 console.println("set properties: " + e.toSource()) }
615         }
616     }
617 }
618 \end{newsegment}
619 \end{acromemory2}
620 \begin{newsegment}{AcroMemory 5: Reporting}
621 function reportProgress(nCorrectAM,nAttemptsAM) {
622     var Msg = this.getField("MsgBox")
623     if ( Msg != null ) {
624         Msg.value = "Number matched = " + nCorrectAM
625             + "\n Number of attempts = " + nAttemptsAM;
626     }
627 }
628 %try { randomizePuzzle(); } catch(e) {}
629 %var to=app.setTimeout("randomizePuzzle();",1000);

```



```
630 \end{newsegment}  
631 \end{insDLJS*}  
632 \end{package}
```

## 5 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
<code>\%</code> .....	305
<code>\@Ext</code> .....	42, 80, 84, 101
<code>\@embedList</code> .....	44, 105, 108
<code>\@isPackagedfalse</code> .....	30, 58, 111
<code>\@isPackagedtrue</code> .....	31
<code>\@rgv</code> .....	159, 189
A	
<code>\AA</code> .....	223
<code>\AAMouseEnter</code> .....	223
<code>\AAMouseExit</code> .....	231
<code>\AAmouseup</code> .....	143, 149, 194, 243, 247
<code>acromemory1 (option)</code> .....	<i>2</i>
<code>acromemory2 (option)</code> .....	<i>2</i>
<code>\acromemoryifalse</code> .....	4, 8
<code>\acromemoryitru</code> .....	3
<code>\allowbreak</code> .....	145, 151, 198
<code>\am@nCnt</code> .....	27, 49–51, 85, 106
<code>\AM@next</code> .....	203, 206, 208, 211
<code>\amEmbedTiles</code> .....	3, 33
<code>\amIconObjs</code> .....	32, 47
<code>\amIconPic</code> .....	5, 113, 135, 143, 149, 182, 196
<code>\amImageHt</code> .....	166, 171
<code>\amImageWd</code> .....	163, 168, 178, 179
<code>\AMIndxList</code> .....	45, 52, 259, 260
<code>\amNumImages</code> .....	34, 47, 48
<code>\amtile@KV</code> s .....	144, 150, 156, 197
<code>\amTileHt</code> .....	124, 145, 151, 173, 198
<code>\amtileKV</code> s .....	156, 157
<code>\amTileWd</code> .....	121, 125, 145, 151, 170, 198
<code>\as@L</code> .....	159, 189
B	
<code>\BC</code> .....	135, 182, 217, 222
<code>\bDebug</code> .....	<u>28</u>
<code>\BG</code> .....	114, 135, 182, 217, 222
<code>\box</code> .....	123, 165
C	
<code>\CA</code> .....	216, 222, 243, 247
<code>\csarg</code> .....	35
<code>\csOf</code> .....	113, 142, 148, 195, 216
D	
<code>\d</code> .....	386
<code>\DeclareOptionX</code> .....	3–6
<code>\dimen</code> .....	215
<code>\divide</code> .....	118, 130, 169, 172
<code>draft (option)</code> .....	<i>2</i>
E	
<code>\embedIcon</code> .....	69, 73, 79, 83, 89, 93, 99
<code>\excludecomment</code> .....	20, 25
<code>\execJSOn</code> .....	10
F	
<code>\F</code> .....	114
<code>\FB</code> .....	143, 149, 196
<code>\Ff</code> .....	217, 241
<code>\FfMultiline</code> .....	241
<code>\FfReadOnly</code> .....	217
<code>\FHidden</code> .....	114
G	
<code>\g@addto@macro</code> .....	46, 52, 104
H	
<code>\hbadness</code> .....	133, 180
<code>\helpCaption</code> .....	216, 228, 238
<code>\helpImage</code> .....	<u>212</u>
I	
<code>\I</code> .....	113
<code>\iconPresets</code> .....	142, 144, 148, 150, 194, 196
<code>\if@isPackaged</code> .....	30, 53, 66, 86
<code>\ifacromemoryi</code> .....	8, 17, 37, 60, 202, 207, 242, 246
<code>\ifincludehelp</code> .....	7, 83, 213, 221
<code>\ifpdf</code> .....	67, 87
<code>\ifxetex</code> .....	13, 53
<code>\imageImportPath</code> .....	36, 214
<code>\includecomment</code> .....	19, 24
<code>\includegraphics</code> .....	119, 161, 214
<code>includehelp (option)</code> .....	<i>2</i>
<code>\includehelpfalse</code> .....	7, 21
<code>\includehelptrue</code> .....	5
<code>\initFirstiiMsg</code> .....	252, 464, 506
<code>\initFirstiMsg</code> .....	250, 378
<code>\insertTiles</code> .....	5, 116
<code>\insertTilesii</code> .....	6, 158, 206, 211
<code>\insertTilesL</code> .....	7, 202, 204
<code>\insertTilesR</code> .....	7, 207, 209
<code>\isPackage</code> .....	3, 31, 56

<code>\IX</code> .....	142, 148, 195, 216	<code>\RanIdentifier</code> .....	18, 23, 293, 317, 342, 345, 362
<b>J</b>			
<code>\JS</code> .....	223, 231	<code>\RequirePackage</code> .....	2, 11, 12, 14–16
<b>M</b>			
<code>\makeXasPDOff</code> .....	13	<code>\rlap</code> .....	135, 182
<code>\memDebug</code> .....	28, 29, 272	<code>\rolloverHelpButton</code> .....	<u>220</u>
<code>\messageBox</code> .....	<u>240</u>	<b>S</b>	
<code>\muAction</code> .....	190, 192, 194	<code>\S</code> .....	114, 217
<code>\multido</code> .....	136, 183	<code>\setbox</code> .....	119, 123, 161, 165, 213
<code>\multiply</code> .....	39, 62, 126, 129, 176	<b>T</b>	
<b>N</b>			
<code>\n</code> .....	625	<code>\textField</code> .....	241
<code>\nTotalTiles</code> .....	40,	<code>\theHelpCaption</code> .....	<u>238</u>
41, 258, 269, 291, 300, 314, 361, 433, 488, 529		<code>\tot@lHalfTiles</code> .....	131, 136
<b>O</b>			
<code>\offinterlineskip</code> .....	133, 180	<code>\Tot@lTiles</code> .....	177, 183
options:		<code>\tot@lWd</code> .....	127, 132
<code>acromemory1</code> .....	2	<code>\TP</code> .....	114, 216
<code>acromemory2</code> .....	2	<code>\tryItAgain</code> .....	246
<code>draft</code> .....	2	<b>V</b>	
<code>includehelp</code> .....	2	<code>\voidb@x</code> .....	123, 165
<b>P</b>			
<code>\PackageWarning</code> .....	54, 203, 208	<b>W</b>	
<code>\PassOptionsToPackage</code> .....	6	<code>\W</code> .....	114
<code>\playItAgain</code> .....	<u>242</u> , <u>246</u>	<b>X</b>	
<code>\presets</code> .....	144, 150, 196, 197	<code>\x</code> .....	50–52, 69, 70, 73, 75, 79, 80,
<code>\ProcessOptionsX</code> .....	9	89, 90, 93, 95, 99, 101, 140–142, 148, 187, 188, 195	
<code>\pushButton</code> .....	114, 216, 222, 243, 247	<b>Y</b>	
<b>R</b>			
<code>\r</code> .....	190, 192, 224–229, 232, 233	<code>\y</code> .....	68, 72, 78, 88,
		92, 98, 105, 138, 145, 147, 151, 185, 191, 193, 198	
		<b>Z</b>	
		<code>\z</code> .....	46, 47, 52, 64, 74, 80, 85, 94, 100
		<code>\zi</code> .....	65, 74, 80

## 6 Change History

v1.1 (2017/02/23)

General: use `f.buttonGetIcon` to get null icon  
object .....

v2.0 (2020/06/23)

General: Rewrote entire package in order to  
support all  $\LaTeX$  workflows .....