# The AcroMemory Package
# A member of the AeB Pro family

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

processed June 26, 2020

## Contents



## 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 LaTeX workflows: pdflatex, lualatex, xelatex, 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 LaTeX package tile-graphic.[1]

```
1 ⟨*package⟩
2 \RequirePackage{xkeyval}
```

`acromemory1` One playing board, where you try to match identical icons.

```
3 \DeclareOptionX{acromemory1}{\acromemoryitrue}
```

`acromemory2` Two playing boards, one board randomized the other not. Try to find the matching 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}
```

[1] https://ctan.org/pkg/tile-graphic

```
  {\RequirePackage[execJS]{eforms}}
\RequirePackage{aeb-comment}
\ifxetex\makeXasPDOff\fi
\RequirePackage{icon-appr}
\RequirePackage{multido}
\RequirePackage{graphicx}
\ifacromemoryi
    \def\RanIdentifier{\@gobble}
    \includecomment{acromemory1}
    \excludecomment{acromemory2}
    \includehelpfalse
\else
    \def\RanIdentifier{R\@gobble}
    \includecomment{acromemory2}
    \excludecomment{acromemory1}
\fi
\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.

```
\def\bDebug{\def\memDebug{true}}
\def\memDebug{false}
```

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

```
\newif\if@isPackaged\@isPackagedfalse
\def\isPackage{\@isPackagedtrue}
\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.

```
\newcommand{\amEmbedTiles}[4][]{\begingroup
  \gdef\amNumImages{#3}%
  \csarg\gdef{amGraphicPath#2}{#4}%
  \gdef\imageImportPath{#4}%
  \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.

```
    \@tempcnta=#3\relax
    \multiply\@tempcnta\tw@
    \xdef\nTotalTiles{\the\@tempcnta}\else
    \gdef\nTotalTiles{#3}\fi
  \def\@Ext{#1}\ifx\@Ext\@empty\def\@Ext{.pdf}\else\def\@Ext{.#1}\fi
  \@tempcnta\z@
```

```
\let\@embedList\@empty
\let\AMIndxList\@gobble
\edef\z{\noexpand\g@addto@macro\noexpand
  \amIconObjs{,"#2":\amNumImages}}\z
\@whilenum \@tempcnta < \amNumImages \do{%
  \am@nCnt\@tempcnta\advance\am@nCnt\@ne
  \ifnum\am@nCnt<10\relax\edef\x{0\the\am@nCnt}\else
    \edef\x{\the\am@nCnt}\fi
  \edef\z{\noexpand\g@addto@macro\noexpand\AMIndxList{,"#2pic\x"}}\z
  \ifxetex\if@isPackaged
    \PackageWarning{acromemory}
    {There is no support for embedding packaged\MessageBreak
    PDFs with xelatex. Ignoring the \string\isPackage\MessageBreak
    command}%
    \@isPackagedfalse
  \fi\fi
  \ifacromemoryi
    \@tempcntb\@tempcnta
    \multiply\@tempcntb\tw@
    \advance\@tempcntb\@ne
    \edef\z{\the\@tempcntb}\advance\@tempcntb\@ne
    \edef\zi{\the\@tempcntb}%
    \if@isPackaged
      \ifpdf
        \edef\y{\noexpand
          \embedIcon[name=#2pic\x,%
          hyopts={page=\x}]{#4_package.pdf}}%
      \else
        \edef\y{\noexpand
          \embedIcon[name=#2pic\x,%
            placement={[1]Membutton.\z,[1]Membutton.\zi},%
            page=\x-1]{#4_package.pdf}}%
      \fi
    \else
      \edef\y{\noexpand
        \embedIcon[name=#2pic\x,%
          placement={[1]Membutton.\z,[1]Membutton.\zi}]{#4_\x\@Ext}}%
    \fi
  \else
    \ifincludehelp\embedIcon[name=helpimage,%
      placement={[1]memoryhelp}]{#4\@Ext}\fi
    \edef\z{\the\am@nCnt}%
    \if@isPackaged
      \ifpdf
        \edef\y{\noexpand
          \embedIcon[name=#2pic\x,%
          hyopts={page=\x}]{#4_package.pdf}}%
      \else
        \edef\y{\noexpand
          \embedIcon[name=#2pic\x,%
```

```
              placement={[1]MemLbutton.\z,[1]MemRbutton.\z},%
            page=\x-1]{#4_package.pdf}}%
        \fi
      \else
        \edef\y{\noexpand
          \embedIcon[name=#2pic\x,%
            placement={[1]MemLbutton.\z,[1]MemRbutton.\z}%
          ]{#4_\x\@Ext}}%
      \fi
    \fi
    \expandafter\g@addto@macro\expandafter
    \@embedList\expandafter{\y}%
    \@tempcnta\am@nCnt
  }% do
  \toks@=\expandafter{\@embedList}\the\toks@
%%\typeout{!! \the\toks@}%
  \endgroup
  \global\@isPackagedfalse
}
```

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

```
\newcommand{\amIconPic}[4][]{% \I{\csOf{name}} required
  \pushButton[\BG{}\W{1}\S{S}#1\TP{1}%\F{\FHidden}
    ]{#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

```
\newcommand\insertTiles[4]{\begingroup
  \@tempdima#2\relax
  \divide\@tempdima #4\relax
  \setbox\z@\hbox{\includegraphics[draft,width=\@tempdima]%
  {\@nameuse{amGraphicPath#1}}}%
  \edef\amTileWd{\the\wd\z@}%
  \setlength\@tempdima{\ht\z@+\dp\z@}%
  \setbox\z@\box\voidb@x
  \edef\amTileHt{\the\@tempdima}%
  \@tempdima\amTileWd\relax
  \multiply\@tempdima #4\relax
  \edef\tot@lWd{\the\@tempdima}%
  \@tempcnta#3\relax
  \multiply\@tempcnta #4\relax
  \divide\@tempcnta\tw@
  \edef\tot@lHalfTiles{\the\@tempcnta}%
```

```
  \begin{minipage}{\tot@lWd}%
    \offinterlineskip\hbadness=10000\@tempcnta\z@
    \leavevmode
    \rlap{\amIconPic[\BC{}\BG{}]{nullIconBtn}{0bp}{0bp}}%
    \multido{\i=1+1}{\tot@lHalfTiles}{%
      \advance\@tempcnta\@ne
      \edef\y{\the\@tempcnta}%
      \ifnum\i<10\relax
        \edef\x{0\i}\else
        \edef\x{\i}\fi
      \edef\iconPresets{\noexpand\IX{\noexpand\csOf{#1pic\x}}}%
      \amIconPic[\AAmouseup{selectTile();}\FB{true}
        \presets{\iconPresets}\presets{\amtile@KVs}
      ]{Membutton.\y}{\amTileWd}{\amTileHt}\allowbreak
      \advance\@tempcnta\@ne
      \edef\y{\the\@tempcnta}%
      \edef\iconPresets{\noexpand\IX{\noexpand\csOf{#1pic\x}}}%
      \amIconPic[\AAmouseup{selectTile();}\FB{true}
        \presets{\iconPresets}\presets{\amtile@KVs}
      ]{Membutton.\y}{\amTileWd}{\amTileHt}\allowbreak
    }% multido
  \end{minipage}%
  \endgroup
}
\def\amtileKVs#1{\def\amtile@KVs{#1}}
\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`.

```
\newcommand\insertTilesii[5]{\begingroup
  \def\@rgv{#5}\def\as@L{L}%
  \@tempdima#2\relax
  \setbox\z@\hbox{\includegraphics[draft,width=\@tempdima]{%
    \@nameuse{amGraphicPath#1}}}%
  \edef\amImageWd{\the\wd\z@}%
  \setlength\@tempdima{\ht\z@+\dp\z@}%
  \setbox\z@\box\voidb@x
  \edef\amImageHt{\the\@tempdima}%
% Now calculate wd and ht of a tile
  \@tempdima\amImageWd\relax
  \divide\@tempdima#4\relax
  \edef\amTileWd{\the\@tempdima}%
```

```
  \@tempdima\amImageHt\relax
  \divide\@tempdima#3\relax
  \edef\amTileHt{\the\@tempdima}%
% Calculate total number of tiles
  \@tempcnta#3\relax
  \multiply\@tempcnta#4\relax
  \edef\Tot@lTiles{\the\@tempcnta}%
% Begin minipage of width \amImageWd
  \begin{minipage}{\amImageWd}%
    \offinterlineskip\hbadness=10000\@tempcnta\z@
    \leavevmode
    \rlap{\amIconPic[\BC{}\BG{}]{nullIconBtn}{0bp}{0bp}}%
    \multido{\i=1+1}{\Tot@lTiles}{%
      \advance\@tempcnta\@ne
      \edef\y{\the\@tempcnta}%
      \ifnum\i<10\relax
        \edef\x{0\i}\else
        \edef\x{\i}\fi
      \ifx\@rgv\as@L
        \def\muAction{nRowsAM=#3;nColsAM=#4;\string\r
          selectNonRandomTile(\y,\y);}\else
        \def\muAction{nRowsAM=#3;nColsAM=#4;\string\r
          selectRandomTile(randomAM[\y],\y);}\fi
      \edef\iconPresets{\noexpand\AAmouseup{\muAction}\noexpand
        \IX{\noexpand\csOf{#1pic\x}}}%
      \amIconPic[\presets{\iconPresets} %\FB{true}
        \presets{\amtile@KVs}
      ]{Mem#5button.\y}{\amTileWd}{\amTileHt}\allowbreak
    }% multido
  \end{minipage}\endgroup
}
```

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

```
\newcommand\insertTilesL[4]{\ifacromemoryi
  \def\AM@next{\PackageWarning{acromemory}
  {The use of \string\insertTilesL\space is supported\MessageBreak
  only for the acromemory2 option}}\else
  \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.

```
\newcommand\insertTilesR[4]{\ifacromemoryi
  \def\AM@next{\PackageWarning{acromemory}
  {The use of \string\insertTilesR\space is supported\MessageBreak
  only for the acromemory2 option}}\else
  \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 it width is set by the command `\setHelpImageWidth`. The image is normally hidden until the user rolls over the `\rolloverHelpButton`. The icons appears with an caption under it, the content of the caption can be entered using `\theHelpCaption`.

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

`\rolloverHelpButton`

```
\newcommand{\rolloverHelpButton}[3][]{%
    \ifincludehelp
        \pushButton[\CA{Help}\BC{0 0 1}\BG{0.89 0.9 0.9}
        \AA{\AAMouseEnter{\JS{%
        var f = this.getField("memoryhelp");\r
        oIcon = f.buttonGetIcon(1);\r
        f.buttonPosition = position.iconTextV;\r
        f.buttonSetIcon(oIcon,0);\r
        f.buttonSetCaption({cCaption: "\helpCaption"});\r
        f.textColor=color.blue;\r
        }}%
        \AAMouseExit{\JS{%
        var f = this.getField("memoryhelp");\r
        f.buttonPosition = position.iconOnly;\r
        f.buttonSetIcon(nullIcon,0);
        }}}#1]{checkhelp}{#2}{#3}%
    \fi
}
```

`\theHelpCaption`

```
\def\theHelpCaption#1{\def\helpCaption{#1}}
\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.

```
\newcommand{\messageBox}[3][]{%
    \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.

```
\newcommand{\playItAgain}[3][]{\ifacromemoryi
    \pushButton[\CA{Play again}#1\AAmouseup{playagain();}]%
    {playAgain}{#2}{#3}\fi
}
```

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

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

# 4 Document JavaScript for AcroMemory

Operational support is provide by JavaScript.

```
\newcommand{\initFirstiMsg}{"Press the 'Play again'
  button to initialize the puzzle"}
\newcommand{\initFirstiiMsg}{"Press the 'Test Your Memory'
  button to initialize the puzzle"}
\begin{insDLJS*}{memjs}
\begin{newsegment}{AcroMemory 1: Global Data and Initialization}
// Global Data:
var isRandomized=false;
var randomAM = new Array(\nTotalTiles+1);
var imageNames = new Array(\AMIndxList);
imageNames.push(\AMIndxList);
imageNames.unshift("null");
var dpsl = randomAM.length;
var timeout = 10;
var shutdown, rAE;
var ok2Continue = true;
var nRowsAM, nColsAM;
var nCorrectAM = 0;
var nAttemptsAM = 0;
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.

```
var f=this.getField("nullIconBtn");
var nullIcon=f.buttonGetIcon();
var debug = \memDebug;
\end{newsegment}
\begin{acromemory1}
\begin{newsegment}{AcroMemory 2: Initialize Pic Names}
var currentChoice = "";
var currentIconName = "";
\end{newsegment}
\end{acromemory1}
\begin{acromemory2}
\begin{newsegment}{AcroMemory 2: Initialize Pic Names}
var LcurrentChoice = 0;
var LcurrentTile = 0;
```

```
var RcurrentChoice = 0;
var RcurrentTile = 0;
\end{newsegment}
\end{acromemory2}
\begin{newsegment}{AcroMemory 3: Bubble Sort}
function clearAM()
{
    for ( var i=1; i<=\nTotalTiles; i++ )
    {
        var f = this.getField("Mem\RanIdentifier button."+i);
        f.buttonSetIcon(nullIcon);
    }
}
function mixupAM()
{
    var i, rand;
    for (i=1; i<= \nTotalTiles; i++)
    {
        var rand = Math.random();
        rand *= dpsl*dpsl;
        rand = Math.ceil(rand);
        rand = rand \% dpsl;
        if (rand == 0 ) rand = 1;
        temp = randomAM[i];
        randomAM[i]=randomAM[rand];
        randomAM[rand]=temp;
    }
}
function showAM()
{
    for ( var i=1; i<=\nTotalTiles; i++ )
    {
        var oIcon = this.getIcon(imageNames[randomAM[i]]);
        var f = this.getField("Mem\RanIdentifier button."+i);
        f.buttonSetIcon(oIcon);
    }
}
// Begin bubble sort
function sortoutAM()
{
    outerLoop(randomAM.length-1);
}
function outerLoop(i)
{
     if ( ok2Continue && (i >= 0) ) shutdown = %
app.setTimeOut("app.clearTimeOut(shutdown); %
innerLoop("+i+",1);", timeout);
}
function innerLoop(i,j)
{
```

```
    if ( j <= i )
    {
       if (randomAM[j-1] > randomAM[j])
       {
            var temp = randomAM[j-1];
            randomAM[j-1] = randomAM[j];
            randomAM[j] = temp;
            var oIcon = this.getIcon(imageNames[randomAM[j-1]]);
            var f = this.getField("Mem\RanIdentifier button."+(j-1));
            f.buttonSetIcon(oIcon);
            var oIcon = this.getIcon(imageNames[randomAM[j]]);
            var f = this.getField("Mem\RanIdentifier button."+j);
            f.buttonSetIcon(oIcon);
        }
        j++
        if ( ok2Continue ) shutdown = %
app.setTimeOut("app.clearTimeOut(shutdown); %
innerLoop("+i+","+j+");", timeout);
    }
    else
    {
        i--;
        outerLoop(i);
    }
}
function randomizePuzzle() {
    mixupAM();
    for ( var i=1; i<=\nTotalTiles; i++) {
        var g = this.getField("Mem\RanIdentifier button."+i);
        var oIcon = this.getIcon(imageNames[randomAM[i]]);
        g.buttonSetIcon(oIcon,1);
        if (debug) g.buttonSetIcon(oIcon,0);
    }
    isRandomized=true;
}
\end{newsegment}
\begin{acromemory1}
\begin{newsegment}{AcroMemory 4: Tile Processing}
var currentIndex="";
var currentName="";
var _bOK1=true;
function selectTile() // right side randomly arranged
{
    if(!isRandomized){
      app.alert(\initFirstiMsg);
      return;
    }
    if (!_bOK1) return;
    var f = event.target;
    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++) {
```

```
        var g = this.getField("Membutton."+i);
        g.buttonSetIcon(nullIcon,0);
    }
    g = this.getField("Membutton");
    g.readonly=false;
    resetCountersAM();
    nCorrectAM = 0;
    nAttemptsAM = 0;
    reportProgress(nCorrectAM,nAttemptsAM);
    randomizePuzzle();
}
\end{newsegment}
\end{acromemory1}
\begin{acromemory2}
\begin{newsegment}{AcroMemory 4: Tile Processing}
// save original positions of fields
var aLRect=new Array();
var aRRect=new Array();
aLRect.push("null");
aRRect.push("null");
var f=this.getField("MemLbutton");
var g=f.getArray();
for (var i=0; i<g.length; i++)aLRect.push(g[i].rect);
var f=this.getField("MemRbutton");
var g=f.getArray();
for (var i=0; i<g.length; i++)aRRect.push(g[i].rect);
var _bOK2=true;
function selectRandomTile(nCnt,n) // right side randomly arranged
{
      if(!isRandomized){
      app.alert(\initFirstiiMsg);
      return;
    }
    if (!_bOK2) return;
    if ( RcurrentChoice != 0 ) return;
    RcurrentChoice = nCnt;
    RcurrentTile = n;
    nAttemptsAM++;
    var f = event.target;
    f.strokeColor = ["RGB", 0, .6, 0];
    var oIcon = f.buttonGetIcon(1);
    f.buttonSetIcon(oIcon,0);
    if ( LcurrentChoice != 0 ) {
        if (debug) console.println(%
"LcurrentChoice = " + LcurrentChoice + ", RcurrentChoice = " %
+ RcurrentChoice);
        if ( LcurrentChoice == nCnt ) {// right answer
            // need to make right side hidden and readonly
            // need to make this button readonly
```

```
            var g = this.getField("MemLbutton."+LcurrentChoice);
            g.strokeColor=color.transparent;
            g.readonly = true;
            f.strokeColor=color.transparent;
            f.readonly = true;
            if (++nCorrectAM == \nTotalTiles ) // game complete
                executePostGameEffects();
            reportProgress(nCorrectAM,nAttemptsAM);
            resetCountersAM();
        } else { // wrong answer
            // need to set current choices back to zero
            reportProgress(nCorrectAM,nAttemptsAM);
            _bOK2=false;
            rAE = app.setTimeOut("resetAfterError(%
"+LcurrentTile+","+RcurrentTile+");_bOK2=true;", 1000);
            resetCountersAM();
        }
    }
}
// left side, arranged in natural order
function selectNonRandomTile(nCnt,n)
{
      if(!isRandomized){
      app.alert(\initFirstiiMsg);
      return;
    }
    if (!_bOK2) return
    if ( LcurrentChoice != 0 ) return;
    LcurrentChoice = nCnt;
    LcurrentTile = n;
    var f = event.target;
    f.strokeColor = ["RGB", 0, .6, 0];
    var oIcon = f.buttonGetIcon(1);
    f.buttonSetIcon(oIcon,0);
    if ( RcurrentChoice != 0 ) {
        if (debug) console.println(%
"LcurrentChoice = " + LcurrentChoice + ", RcurrentChoice = " %
+ RcurrentChoice);
        if ( RcurrentChoice == nCnt ) {// right answer
            // need to make right side hidden and readonly
            // need to make this button readonly
            var g = this.getField("MemRbutton."+RcurrentTile);
            g.strokeColor=color.transparent;
            g.readonly = true;
            f.readonly = true;
            f.strokeColor=color.transparent;
            if (++nCorrectAM == \nTotalTiles ) // game complete
                executePostGameEffects();
            reportProgress(nCorrectAM,nAttemptsAM);
```

```
            resetCountersAM();
        } else { // wrong answer
            // need to set current choices back to zero
            reportProgress(nCorrectAM,nAttemptsAM);
            _bOK2=false;
            rAE = app.setTimeOut(%
"resetAfterError("+LcurrentTile+","+RcurrentTile+");%
_bOK2=true;", 1000);
            resetCountersAM();
        }
    }
}
function resetCountersAM ()
{
    LcurrentChoice = 0;
    RcurrentChoice = 0;
    LcurrentTile = 0;
    RcurrentTile = 0;
}
function resetAfterError(l,r)
{
    try { app.clearTimeOut(rAE); } catch(e) {};
    var f = this.getField("MemLbutton."+l);
    var g = this.getField("MemRbutton."+r);
    if (!debug) g.buttonSetIcon(nullIcon,0);
    g.strokeColor=color.black;
    if (!debug) f.buttonSetIcon(nullIcon,0);
    f.strokeColor=color.black;
}
function tryAgain() {
  nCorrectAM=0;
  nAttemptsAM=0;
  reportProgress(nCorrectAM,nAttemptsAM);
%  this.resetForm("MsgBox");
  resetCountersAM();
  this.delay=true;
  for(var i=1; i<=20; i++) {
    var f=this.getField("MemLbutton."+i);
    var g=this.getField("MemRbutton."+i);
    f.buttonSetIcon(nullIcon,0);
    f.rect=aLRect[i];
    g.buttonSetIcon(nullIcon,0);
    g.rect=aRRect[i];
  }
  var f=this.getField("MemLbutton");
  var g=this.getField("MemRbutton");
  f.lineWidth=1;
  f.strokeColor=color.black;
  f.readonly=false;
```

```
  g.lineWidth=1;
  g.strokeColor=color.black;
  g.readonly=false;
  this.delay=false;
  randomizePuzzle();
}
function executePostGameEffects() {
    sortoutAM();
    var fL = this.getField("MemLbutton.1");
    var fR = this.getField("MemRbutton.1");
    var LulCorner = fL.rect;
    var RulCorner = fR.rect;
    var mWidth = LulCorner[2]-LulCorner[0];
    var mHeight = LulCorner[1]-LulCorner[3];
    var nCnt = 0;
    for ( var i=0; i<nRowsAM; i++) {
        for ( var j=0; j<nColsAM; j++ ) {
            nCnt++;
            try {
                var g = this.getField("MemLbutton."+nCnt);
                g.rect  = %
[ LulCorner[0]+j*mWidth, LulCorner[1]-i*mHeight, %
LulCorner[0]+(j+1)*mWidth, %
LulCorner[1]-(i+1)*mHeight ]
                g.lineWidth = 0;
                g.strokeColor = color.transparent;
                var h = this.getField("MemRbutton."+nCnt);
                h.rect  = [ RulCorner[0]+j*mWidth, %
RulCorner[1]-i*mHeight, %
RulCorner[0]+(j+1)*mWidth, RulCorner[1]-(i+1)*mHeight ]
                h.lineWidth = 0;
                h.strokeColor = color.transparent;
            } catch(e) { %
console.println("set properties: " + e.toSource()) }
        }
    }
}
\end{newsegment}
\end{acromemory2}
\begin{newsegment}{AcroMemory 5: Reporting}
function reportProgress(nCorrectAM,nAttemptsAM) {
    var Msg = this.getField("MsgBox")
    if ( Msg != null ) {
    Msg.value = "Number matched = " + nCorrectAM
        + "\n Number of attempts = " + nAttemptsAM;
    }
}
%try { randomizePuzzle(); } catch(e) {}
%var to=app.setTimeOut("randomizePuzzle();",1000);
```

```
\end{newsegment}
\end{insDLJS*}
⟨/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.

# 6 Change History

v1.1 (2017/02/23)
General: use f.buttonGetIcon to get null icon object . . . . . 9

v2.0 (2020/06/23)
General: Rewrote entire package in order to support all LaTeXworkflows . . . . . 2