Image Memory Test

Instructions - June 2016

How to install

The software has been tested in Linux and Windows. While Python should offer a platform independent environment, it may NOT work on other systems.

1. Verify that you have a working Python environment (and install it, if the need be).
   Be ready to adapt the instructions for your operative system.

   Open a command prompt (Windows 7: type `cmd` in the search box in the Start menu; Windows 8.1: right-click on the Windows menu button; Linux: your usual terminal) and run the command `python --version`.

   On Windows, download the latest release from https://www.python.org/downloads/ (for example, 2.7.11). Double click on the installer file, be sure to select the option to *include Python in the PATH*.

   You should not need to install Python again if you install a new version of Image Memory Test.

2. Un-zip your copy of the software in any folder.

3. Install the required Python packages.
   Linux: `python -m pip install -r requirements.txt`
   Windows: double click on install.bat

   Avast (and maybe other anti-virus softwares) may block the installation. Disable the anti-virus for the duration of the installation.

How to use

Launch the program with a double click on `MemoryTest.bat` (Windows only) or via the command line (`python main.py`).

Then select an experiment file (a text file which describe the images to show, for how long, where to save the results…) and let the program run.

To "shortcut" the file selection window, you can select the experiment file via the -e command line option (`python main.py -e some/file.txt`). On Windows, make a copy of `MemoryTest.bat` and open it with a text editor (Notepad). Add the option in the file, then save and double click on it.

The experiment starts immediately after the file selection. See below how to customize the experiments (e. g. to wait for a key press or a fixed time, how to change the images and their display times...)

Press the ESC key at any time to close the program.

How to write an experiment file

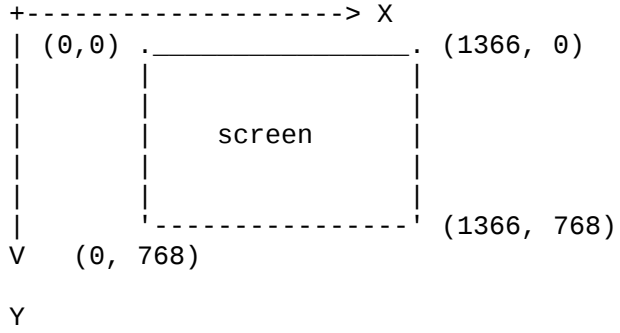
The experiment file describes the sequence of actions (display images, markers etc.) to be performed during the experiment.

Create the file with a plain text editor (Notepad, Notepad++, Gedit… NOT Microsoft Word).
Write each command on a line, in the order they should be executed.
Each command has a name and some parameters between parenthesis, see the list below. For example "Wait(800)" instructs the software to pause for 800 milliseconds.

The program ignore as comment any text following a # sign, up to the end of the line.
The program **halts with an error message** if there are wrong commands.

See the examples in the "Demo" folder.

All times are expressed in milliseconds.
All positions are in the screen reference system. The **origin (0, 0) is on the top left screen corner. X grows to the left, Y grows down. The coordinates units are screen pixels.**



The maximum coordinates may vary according to the screen resolution

Some commands takes file names as parameters. All file paths are relative to the position of the experiment file itself. Example: "anotherFile.jpg" implies that anotherFile.jpg is in the same folder as the experiment file; "dir/anotherFile.jpg" search for the file in the dir folder, itself at the same level as the experiment file.

File paths and text string values should be between single quotes ('like this') or double quotes ("this way").

Numbers can be computed with expression (e. g. wait 20 minutes: Wait(20 * 60 * 1000)).

Images must be in Jpeg format (.jpg extension) and are never resized.

List of commands

`ChangeBackgroundColor(r, g, b)`
: Change the color of the screen to the color expressed by the given red green, blue triplet (https://en.wikipedia.org/wiki/RGB_color_model). R, g, b must be integers between 0 and 255 (included).

`PressKeyToContinue(key_to_wait)`
: Pauses the program until the user presses the given key (e. g. “a”). Use “<Return>” for the enter key, “<space>” for the space bar and “<any>” to continue on any key.

`Wait(milliseconds)`
: Does nothing, just waits the specified amount of time.

`ShowInstructions(key_to_continue, instruction_image_file)`
: Show an image that is not accounted for the memory test. It is meant to give instructions to the text subject. Pauses the experiment until the user presses the key_to_continue key. The image is at the centre of the screen.

`ShowImage(how_long_milliseconds, centre_x_pixels, centre_y_pixels, filename)`
: Loads the image in the file and displays it for the given amount of time. The image is positioned on the screen so that its centre is at the specified coordinates (+/- 1 pixel in each axis if the image side sizes are odd). The program halts if the image does not entirely fit on the screen.

`ShowAllImages(for_how_long)`
: Displays at the same time all the images specified by ShowImage commands, for the duration given as parameter (in milliseconds).

`AllImagesAsBackground()`
: Displays at the same time all the images specified by ShowImage commands. Does not wait.

`RemoveBackgroundImages()`
: Immediately removes all the images placed by AllImagesAsBackground().

`HideBackgroundImage(for_how_long, filename)`
: Removes from the screed the image opened by ShowImage with the same filename and displayed with AllImagesAsBackground for the given time. Then it shows it again.

`ShowConfiguration(for_how_long, marker_image)`
: Shows the images’ configuration: places a copy of marker_image in the position of each image declared with a ShowImage command. Shows all these markers, pauses executions for_how_long milliseconds and leaves the marker in place after.

`EraseConfiguration()`
: Immediately erases the markers placed by ShowConfiguration.

`PrepareMarkers(marker_image, timing_option)`
: Creates a row of markers (one per each image loaded by a ShowImage command) at the bottom of the screen for the user to move (marker_image is a file path). Timing_option can be either of (including the

quotes) "immediate timing" or "time on marker movement". The former starts the experiment duration chronometer as soon as the markers are read, the latter as soon as the user moves a marker. The program halts if the markers don't fit in the last screen row (e. g. there are too many).

If the marker image is the special "<with images>" expression, then the markers will be the images themselves, rather than a specific marker image.

`ComputeResult(result_file)`

Completes the experiment. Computes the distance between each image and the closest marker and the total duration of the experiment. Writes the results in result_file. Allows to select a file with the usual "save us" window if result_file is the special value "<SaveAs>".

The distance computing assumes that the test subject won't be "too far off". The program may give nonsensical results in some corner cases (e. g. if all the markers are clustered around the same position, or if one or more markers are exactly at the same distance with respect to an image).

The timer truncates to a granularity of seconds (but starts and stop time are rounded – there may be small discrepancies).

`CoverImage(for_how_long, image_to_hide, covering_image)`

Displays the covering image on top of the image to hide, then removes it after for_how_long milliseconds. The covering image has the same center of the image to hide. If the image to hide has to be perfectly covered, then the two images must have the same size.