NIMH MonkeyLogic 2: Open source experimental control and data acquisition

Jaewon Hwang, Andrew R. Mitz, and Elisabeth A. Murray

Section on the Neurobiology of Learning and Memory, Laboratory of Neuropsychology, National Institute of Mental Health, NIH, Bethesda, MD

532.07

SfN 2017

INTRODUCTION

- MonkeyLogic (ML1) is an open source MATLAB toolbox for behavioral control and data acquisition (Asaad & Eskandar, 2008). To date, it has been cited in >50 peer-reviewed publications and is used by >30 research groups worldwide.
- However, ML1 does not support the 64-bit computing environment and has incompatibility issues with MATLAB's new HG2 graphics engine. In addition, ML1 requires the MATLAB Data Acquisition Toolbox and two identical interface boards to operate at full speed.
- We now report the development of NIMH MonkeyLogic 2 (ML2), which resolves those key issues, reduces cost, improves close-loop performance and adds other new features.

SYSTEM REQUIREMENTS

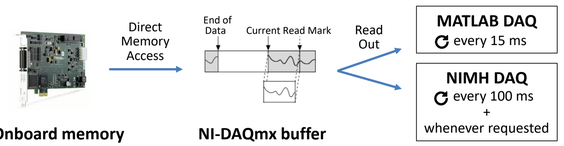
- MATLAB R2011a or later (tested up to R2017b)
  - No MATLAB toolbox is required.
- Windows 7 or later
- Microsoft Visual C++ 2013 Redistributable & DirectX 9.0c runtime (download from MS website)
- National Instruments DAQ board (optional)
  - No need to install two boards
  - USB-type devices are supported.

KEY FEATURES OF ML2

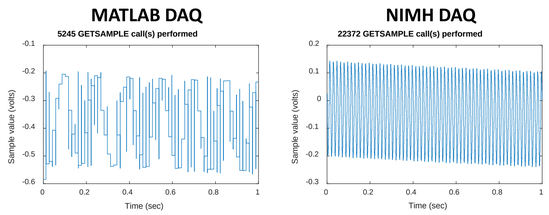
- Full support for latest 64-bit MATLAB
- Compatible with ML1 behavioral tasks
- NIMH DAQ Toolbox (NIMH DAQ)
  - Near real-time behavior monitoring (1-ms resolution) using only one DAQ board
  - Support for more input devices, including mouse/touch screen and USB joystick, in addition to analog and digital devices
- MonkeyLogic Graphics Library (MGL)
  - "What you see is what your monkey sees."
  - Support for transparent images by alpha blending or color key
  - Movie streaming (no limit to movie length)
  - Low-latency audio output with XAudio2
- mlplayer: a trial-replay and video-exporting tool
- Simulation mode that allows testing user tasks with no special hardware

NIMH DAQ TOOLBOX

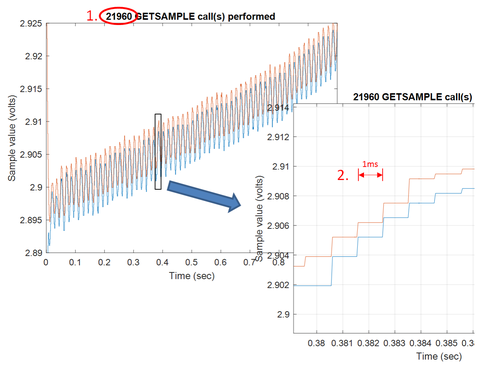
- NIMH DAQ fetches new samples both synchronously and asynchronously.



- During the continuous acquisition mode, newly acquired samples are immediately available in NIMH DAQ.

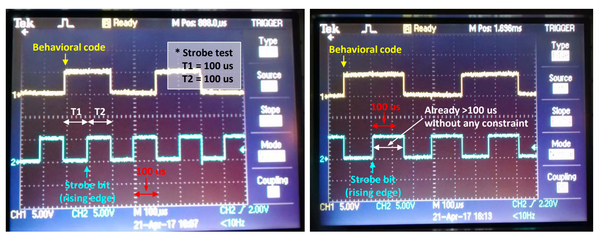


- This fast sample update enables ML2 to perform real-time behavior monitoring with one DAQ board.



- NIMH DAQ bypasses slow m-files and handles time-critical jobs at the DLL level, to improve timing precision.

ML2 with NIMH DAQ | **MATLAB DAQ**

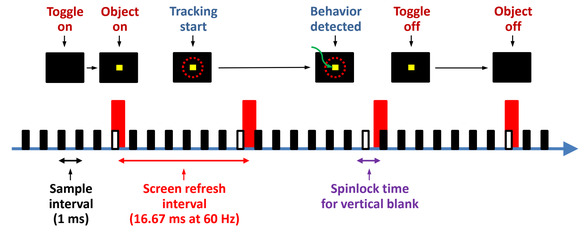


Strobe test details. For NIMH DAQ, the duration of the strobe bit was timed so that T1 and T2 can be 100 us, respectively. For MATLAB DAQ (R2014a, 32-bit), the strobe bit was not timed and the pulse was turned off as soon as it was turned on.

NEW RUNTIME FOR VISUAL EXP

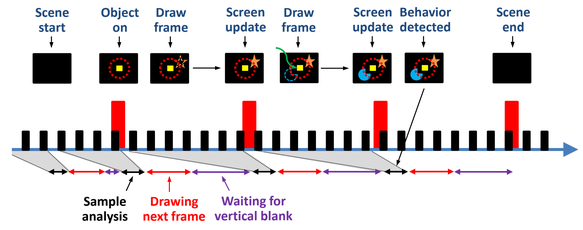
- ML has used two separate functions, toggleobject() and eyejoytrack(), to present stimuli and track behavior, respectively. This structure is disadvantageous in designing behavior-responsive stimuli and interactive experiments.

```
toggleobject(taskobject, 'eventmarker', eventcode);
eyejoytrack('acquirefix', taskobject, threshold, duration);
```



- ML2 provides a new way to compose tasks (runtime v2). In this new runtime, samples collected during one refresh interval are analyzed at the beginning of the next interval to determine what to present.

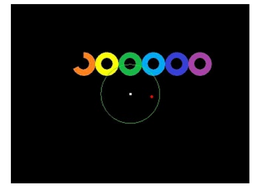
```
scene = create_scene(adapter, taskobject);
run_scene(scene, eventcode);
```



- The "adapter" is a MATLAB class and has two member functions, analyze() and draw(), that are called by run_scene() every frame. Multiple adapters can be concatenated to create complex stimuli or detect complex behavioral patterns.

```
classdef ADAPTER_TEMPLATE < handle
    properties (Access = protected)
        Adapter  % low-level adapter
        Success  % status of behavior detection
    end
    methods
        function obj = ADAPTER_TEMPLATE(adapter)      % constructor
            obj.Adapter = adapter;     % create a chain of adapters
        end
        function continue_ = analyze(obj,p)     % sample analysis
            continue_ = obj.Adapter.analyze(p); % call next adapter
            obj.Success = obj.Adapter.Success;
        end
        function draw(obj,p)                    % drawing next frame
            obj.Adapter.draw(p);                % call next adapter
        end
    end
end
```

EXAMPLE TASK: TimerDemo



- **Adapter composition**
  - TimerDemo — Count the fixation time and update graphics
  - SingleTarget — Check eye fixation
  - EyeTracker — Collect eye signal samples

- Complex behavior and dynamic stimuli
  - The timer counts only while the fixation is held.
  - A trial ends when the timer counts up to 7 s or when the gaze is away for more than 4 s.
- Timing file

```
fix = SingleTarget(eye_);     % EyeTracker, eye_, is pre-defined
fix.Target = [0 0];           % Position to fixate, in degrees
fix.Threshold = 3;            % Fixation window, in degrees
timer = TimerDemo(fix);
scene = create_scene(timer);
run_scene(scene);
```

- TimerDemo adapter (pseudo-code)

```
function continue_ = analyze(obj,p)
    obj.Adapter.analyze(p);  % Call SingleTarget's analyze()
    if obj.Adapter.Success   % If fixation is held,
        TotalFixationTime = TotalFixationTime + ElapsedTime;
        MaxFixationBreak = CurrentTime + 4;  % Time unit is sec
    end
    obj.Success = 7 < TotalFixationTime;
    continue_ = ~obj.Success && MaxFixationBreak < CurrentTime;
end
function draw(obj.p)
    obj.Adapter.draw(p);     % Call SingleTarget's draw()
    if 0 < TotalFixationTime
        sec = floor(TotalFixationTime);   % Which pie to update
        msec = rem(TotalFixationTime, 1); % Center angle of pie
        mglactivategraphic(PieID(:,sec+1),true);
        mglsetproperty(PieID(1,sec+1), 'angle', 360 * msec);
        % There is a small black circle on each pie to make the
        % whole shape look like a doughnut. We don't update it.
    end
end
```

SUMMARY

- ML2 is less expensive to implement than ML1, has substantially improved timing precision and graphics, and supports more input devices.
- Google "NIMH MonkeyLogic 2" and visit the MonkeyLogic Forum, for more information.

REFERENCES / ACKNOWLEDGEMENTS

- Asaad WF, Eskandar EN (2008) Achieving behavioral control with millisecond resolution in a high-level programming environment. *J Neurosci Methods* 173:235-240.
- Asaad WF, Eskandar EN (2008) A flexible software tool for temporally-precise behavioral control in MATLAB. *J Neurosci Methods* 174:245-258.

- We thank Wael Asaad, David Freedman and Edward Ryklin for their support throughout this project.